Robotic arm using MATLAB

By Morteda Hokman Saya

Outlines

- Introduction
- Project Overview
- Components & Tools
- Implementation
- Results
- Tunning
- Results after Controller
- Demonstration
- Challenges & Solutions
- Future Improvements
- Conclusion
- Reference's

Introduction

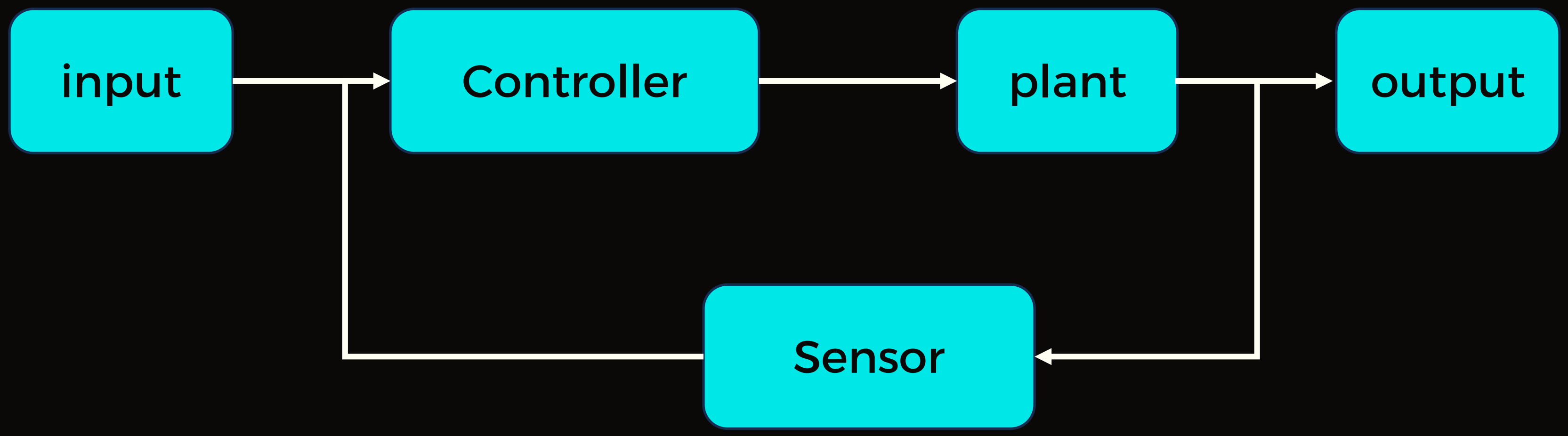
- This robotic arm has 2 joints is designed to be controlled seamlessly using MATLAB. By simply inputting a desired angle, the arm precisely moves to that position, ensuring accurate and efficient operation. Its functionality mirrors the robotic systems commonly used in factories and commercial settings, making it a practical solution for automation and precision tasks.

Why we don’t Use a stepper Motor?
What is advantage of DC motor with encoder?

- **Higher Speed & Torque** – DC motors are faster and often provide more torque than stepper motors.
- **Closed-Loop Control** – The encoder continuously sends position data, allowing corrections if there is an error.
- **Energy Efficient** – Uses power only when needed, reducing energy consumption.
- **Smooth Motion** – Avoids the jerky movement that can occur with stepper motors.

Project Overview

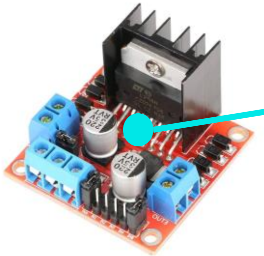
- After the desired angle is entered, the signal is sent to the controller, which processes the input and transmits it simultaneously to the interface and the display screen for real-time monitoring. The processed signal is then sent to the plant, where the system executes the movement. Any feedback from the plant is returned to the controller for error detection and correction, ensuring precise and accurate positioning.



Components & Tools

- Hardware

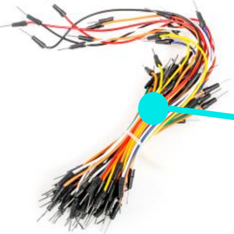
Arduino Mega 2560
26,000 IQD



L298N Motor Driver
4,000 IQD

2x

JGA25-370 DC MOTOR
17,500 IQD



Jumper Wires
2,000 IQD

12v EU Plug Adapter
3,750 IQD

Total Cost
70,750 IQD

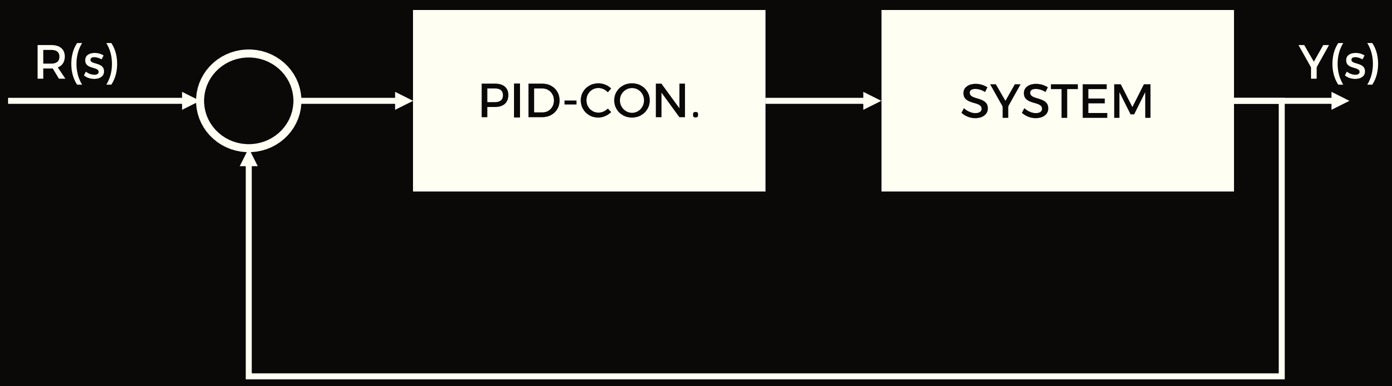
Components & Tools

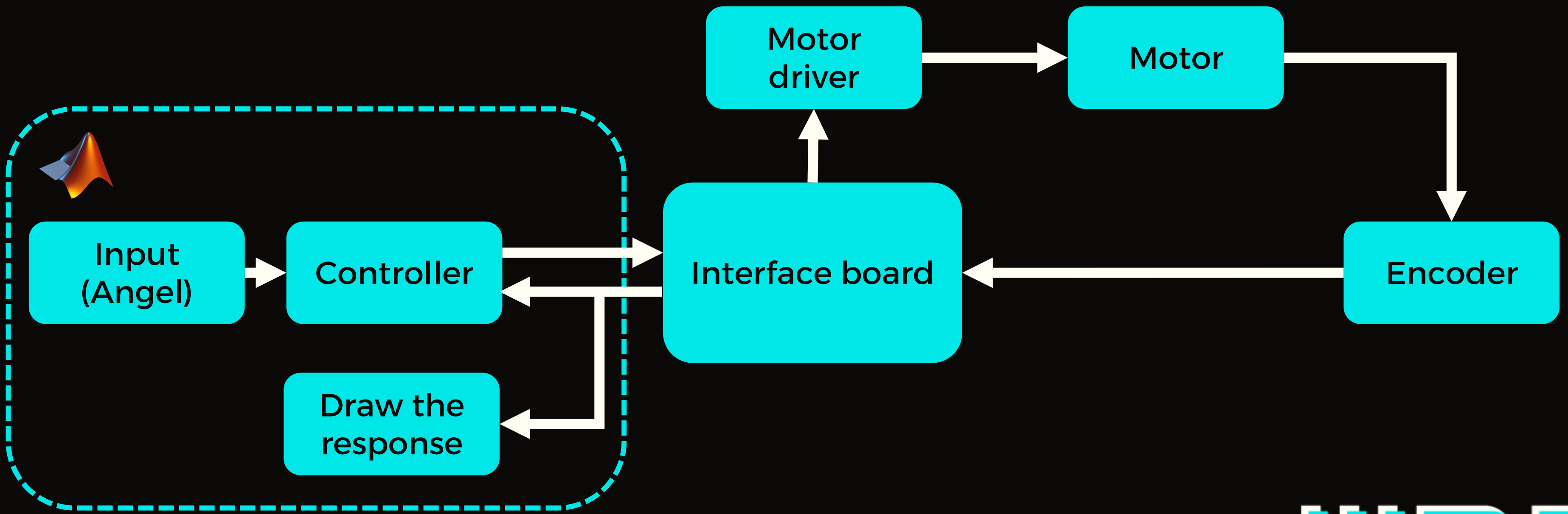
- Software

R2024b
Release Highlight

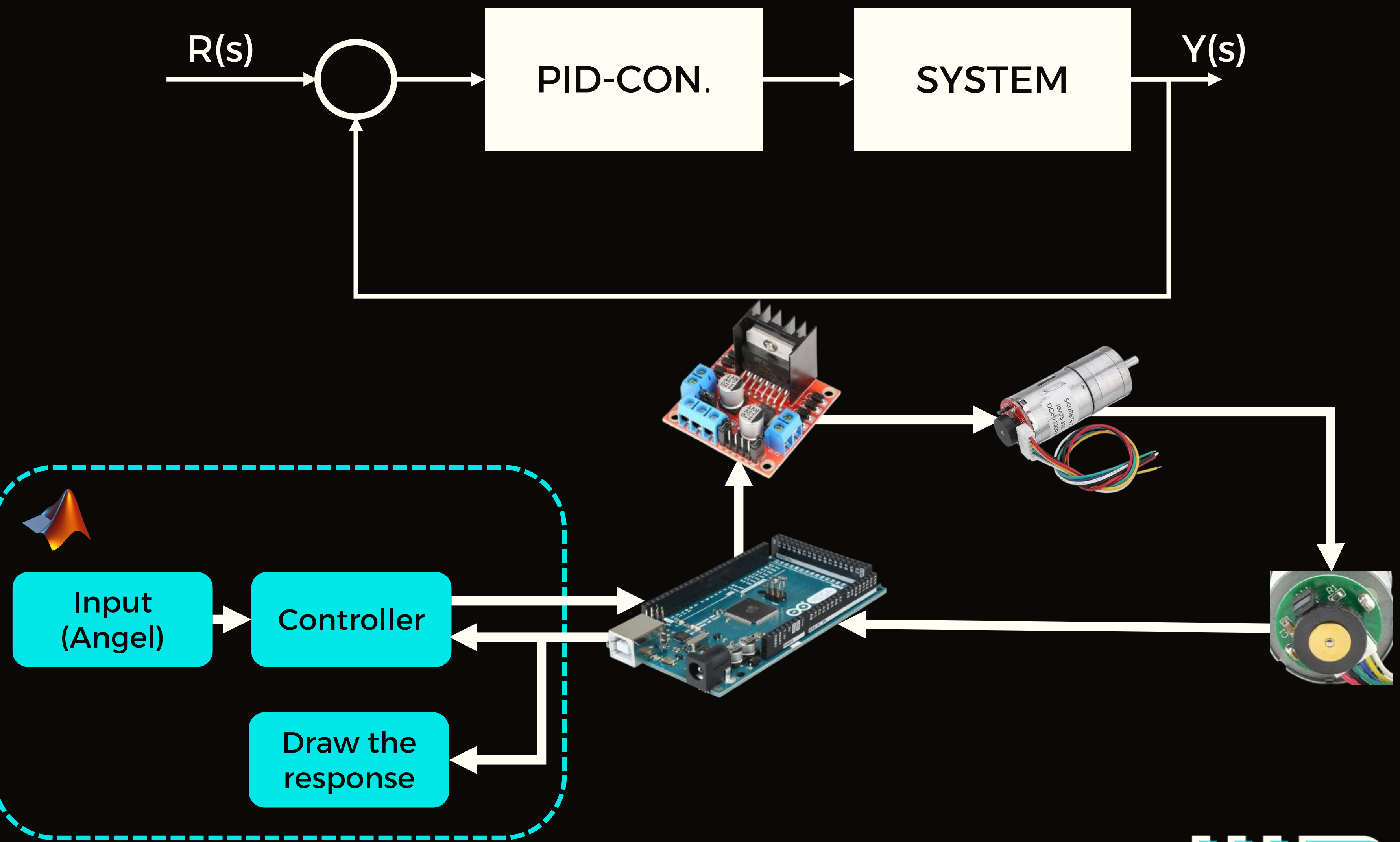
Operating System	• Windows 11 • Windows 10 (version 21H2 or higher) • Windows Server 2022 **Note**: • Windows Server 2019 is no longer supported
Processor	• Minimum: Any Intel or AMD x86-64 processor with two or more cores • Recommended: Any Intel or AMD x86-64 processor with four or more cores and AVX2 instruction set support **Note**: • A future release of MATLAB will require a processor with AVX2 instruction set support
RAM	• Minimum: 8 GB • Recommended: 16 GB
Storage	• 3.8 GB for just MATLAB • 4-6 GB for a typical installation • 23 GB for an all products installation • An SSD is strongly recommended
Graphics	• No specific graphics card is required, but a hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended. • GPU acceleration using Parallel Computing Toolbox requires a GPU with a specific range of compute capability. For more information, see *GPU Computing Requirements*.

Implementation(Hardware)





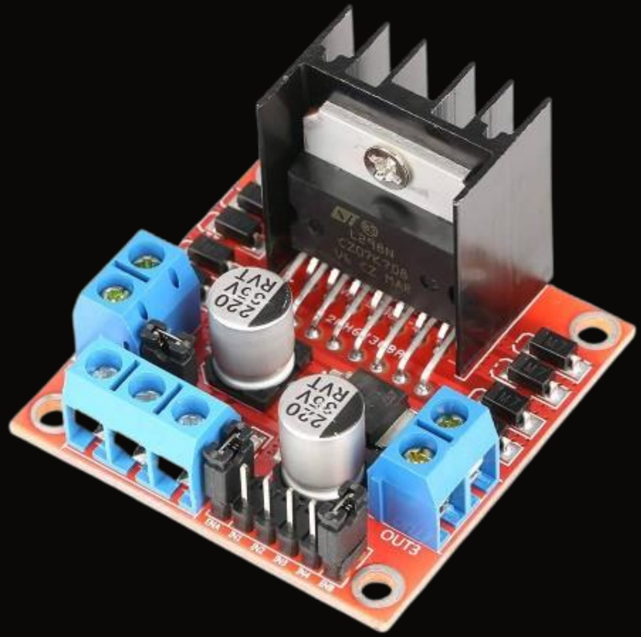
Implementation(Hardware)



Implementation(Hardware)

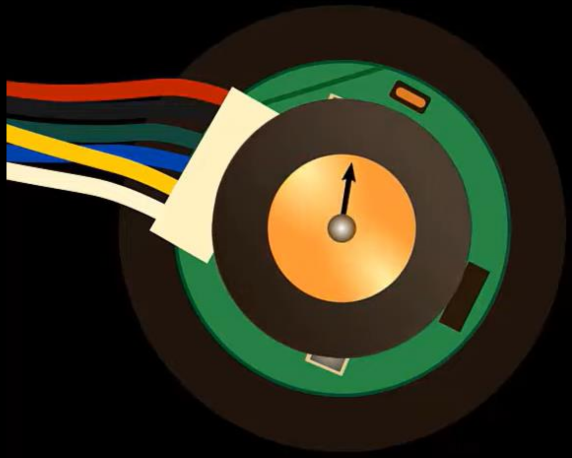
L298N DC Motor Driver – How It Works

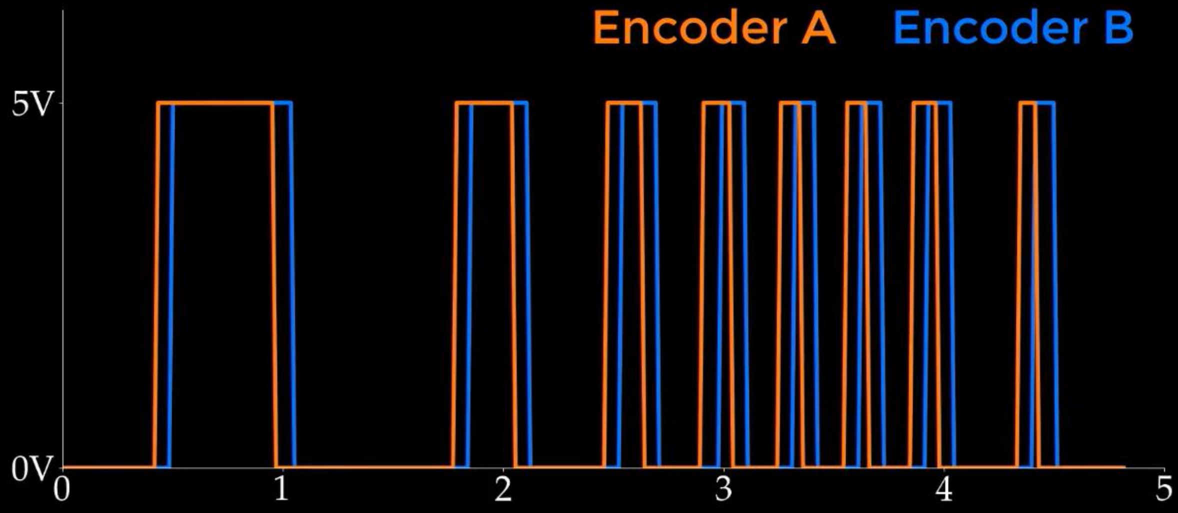
- L298N 2A Based Motor Driver is a high power motor driver perfect for driving DC Motors and Stepper Motors
- It uses the popular L298 motor driver IC and has an onboard 5V regulator which it can supply to an external circuit. It can control up to 4 DC motors, or 2 DC motors with directional and speed control.
- This motor driver is perfect for robotics and mechatronics projects and perfect for controlling motors from microcontrollers, switches, relays, etc. Perfect for driving DC and Stepper motors for micro mouse, line following robots, robot arms, etc.



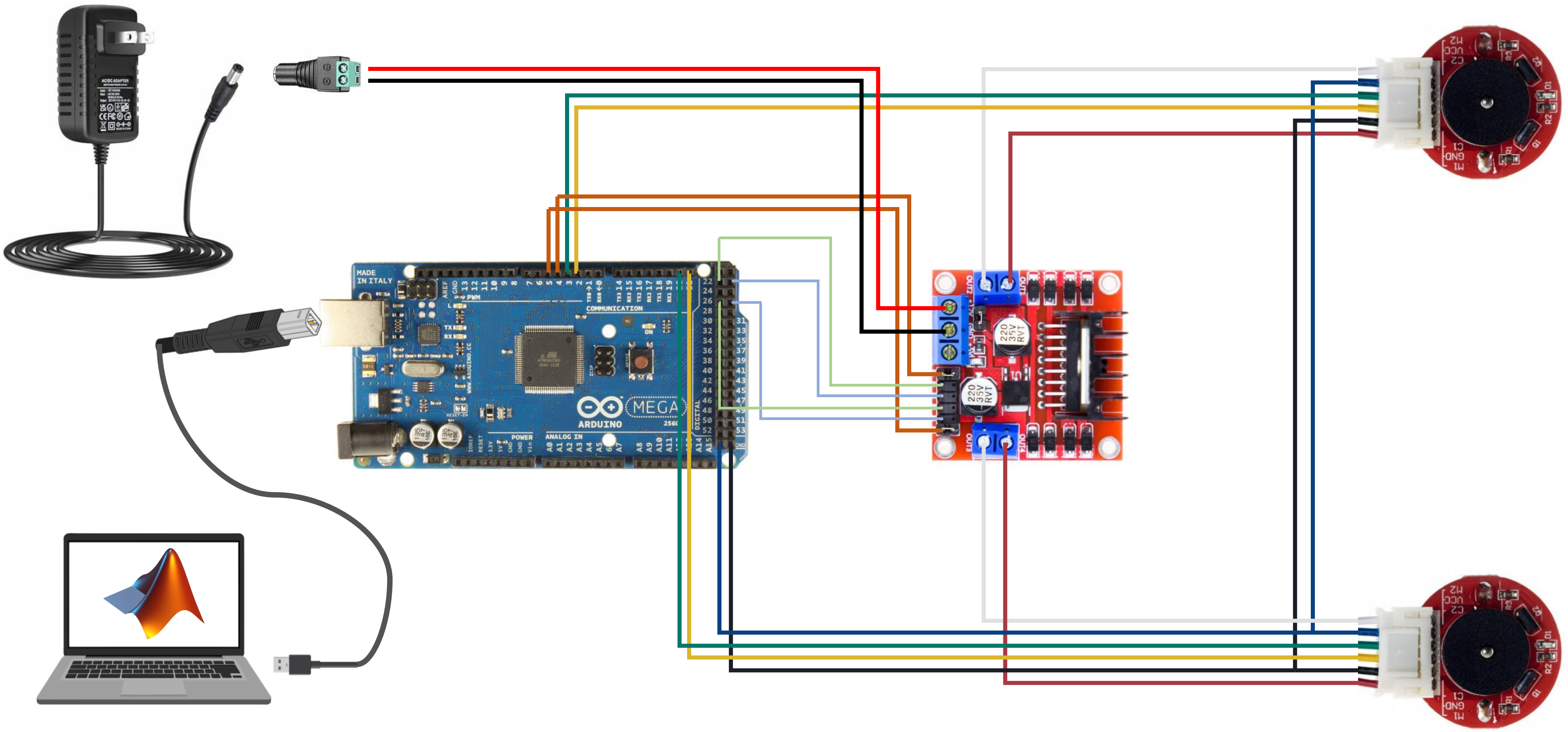
Implementation(Hardware)

JGA25-370 DC GEAR MOTOR with magnetic encoder – How It Works



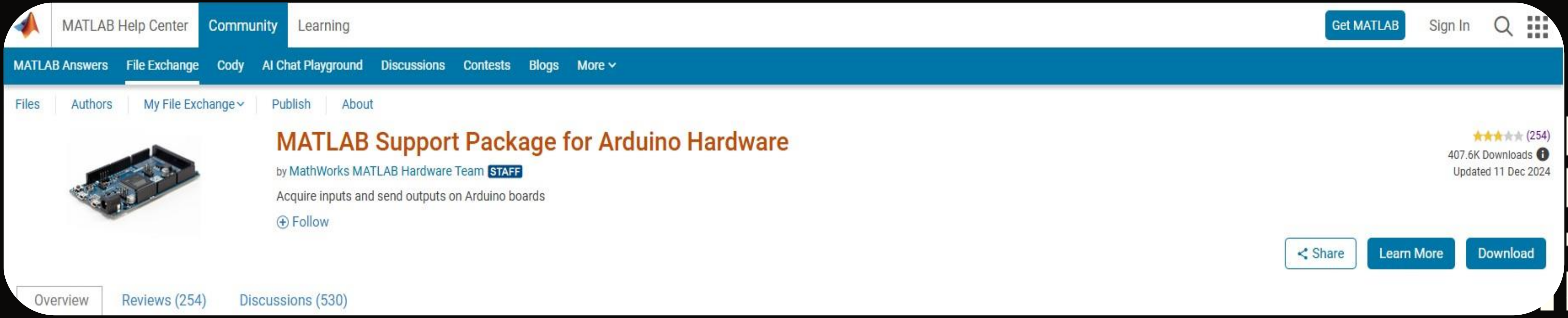


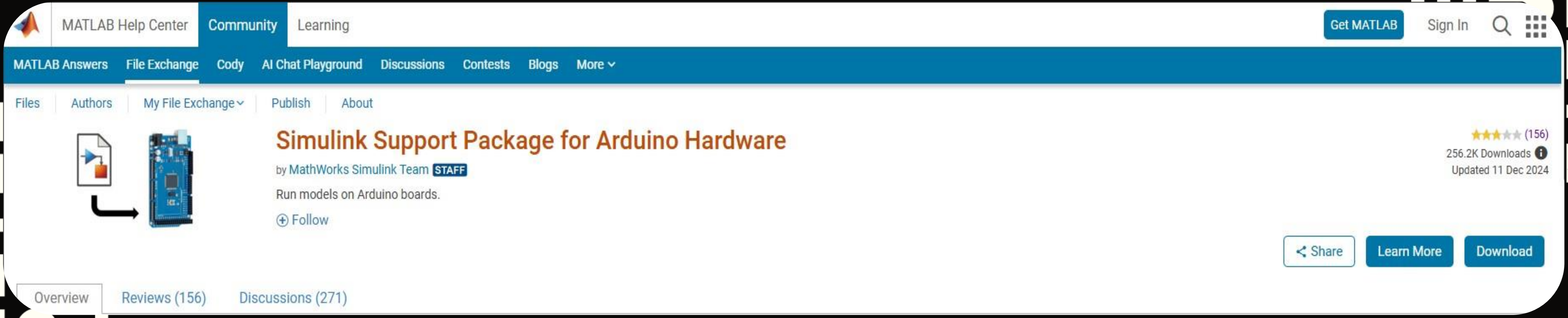
Implementation(Hardware)



Implementation(Software)

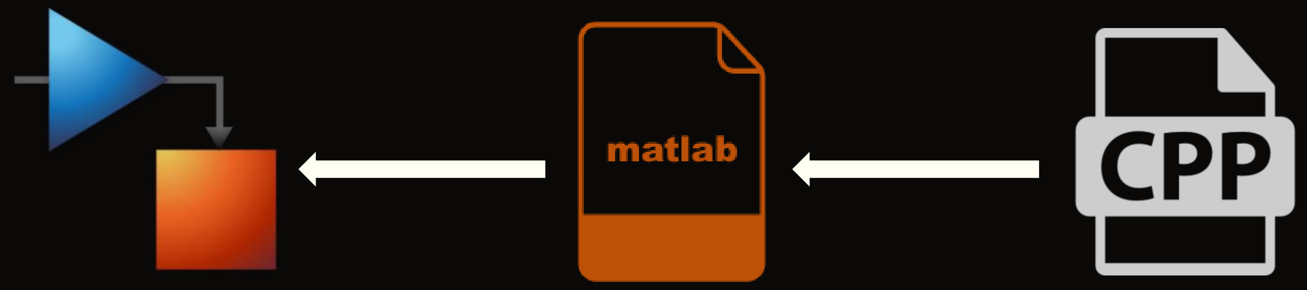
Download Arduino library





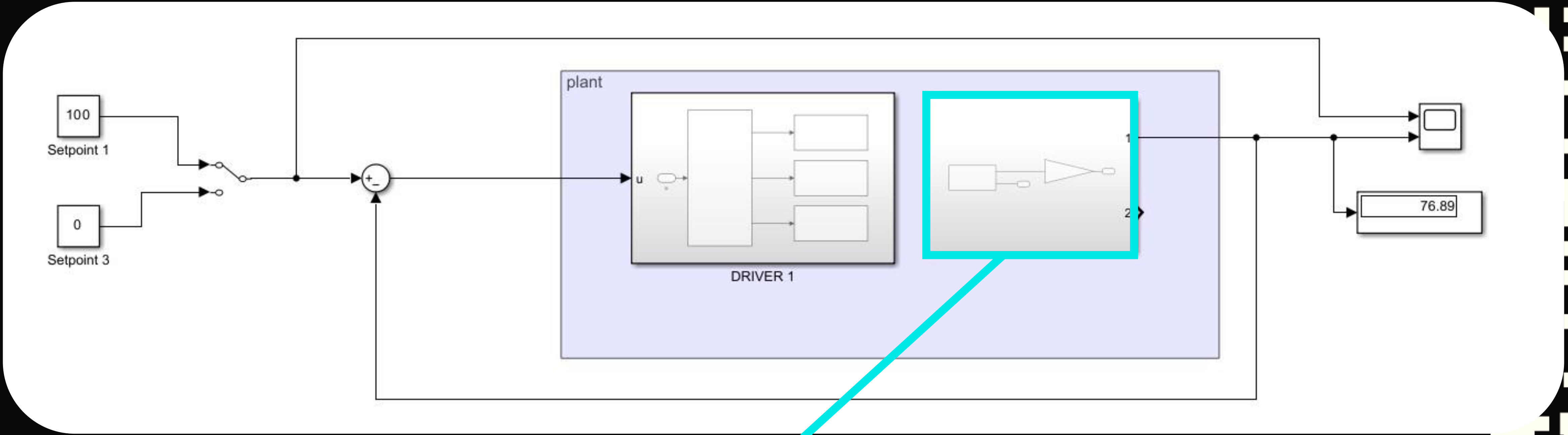
Implementation(Software)

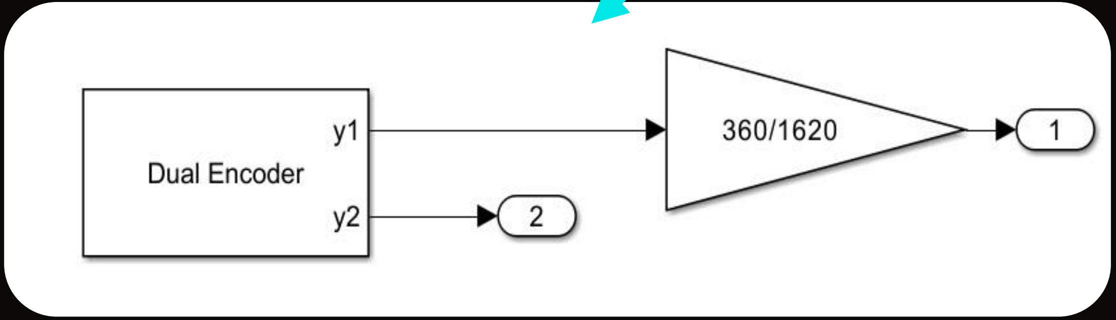
Read form Encoder



Implementation(Software)

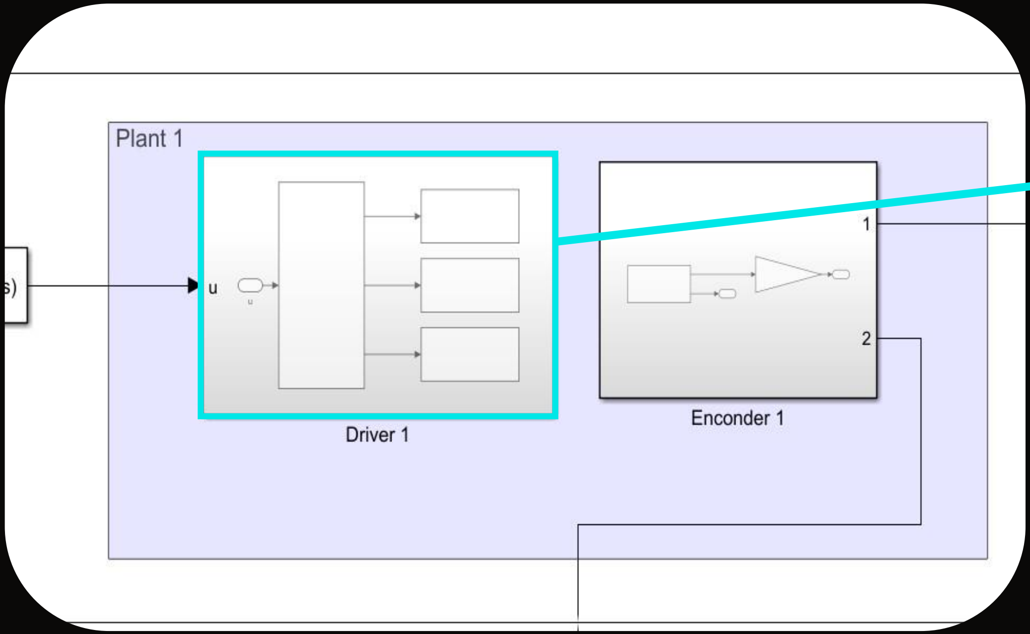
Simulink connection

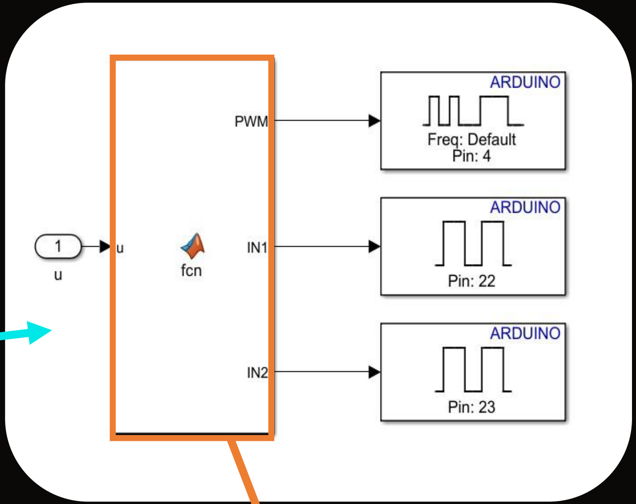




Implementation(Software)

Simulink connection





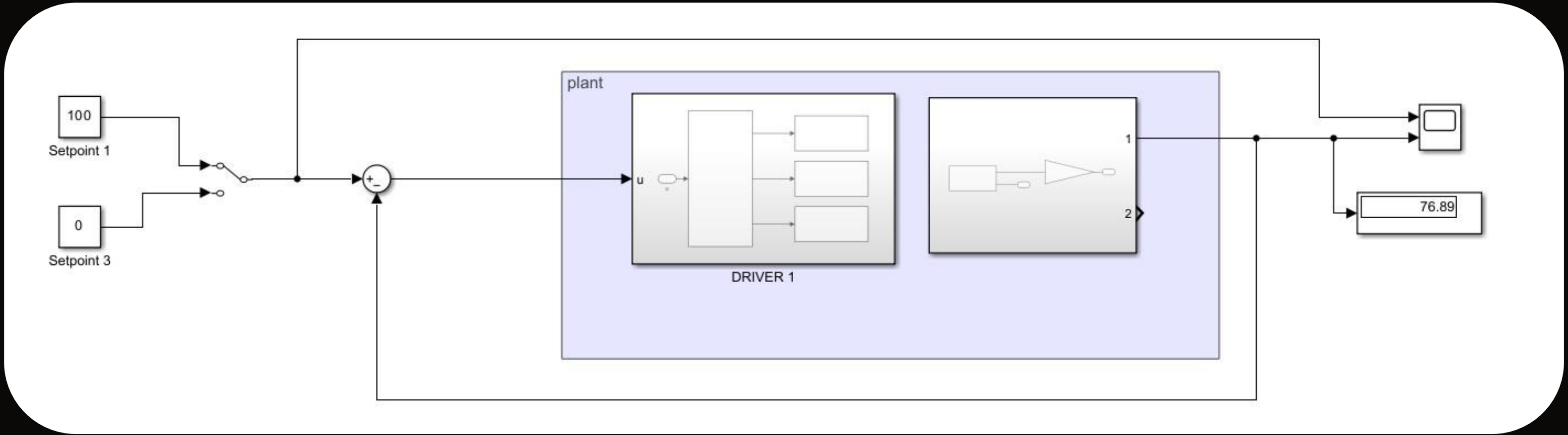
encoder_arduino_test ▸ Driver 1 ▸ MATLAB Function

```matlab
function [PWM,IN1,IN2] = fcn(u)

PWM = abs(u);
if u>0
   IN1 = 1; IN2 =0;
elseif u<0
   IN1 = 0; IN2=1;
else
   IN1=0; IN2=0;
end
```

Results

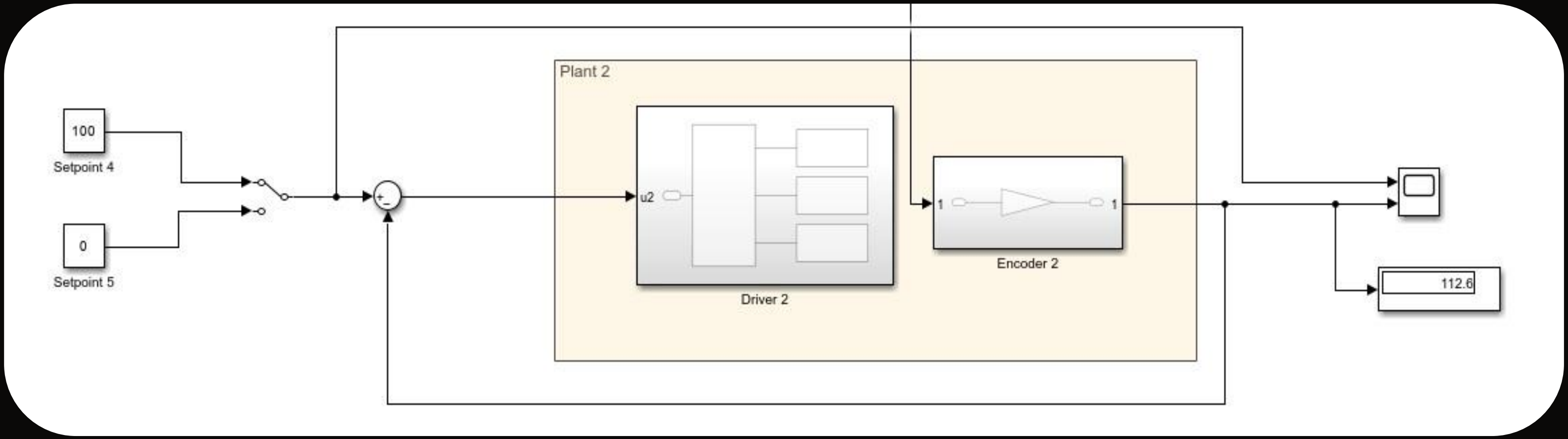
Connection and output of dc Motor Num. 1 (200RPM)

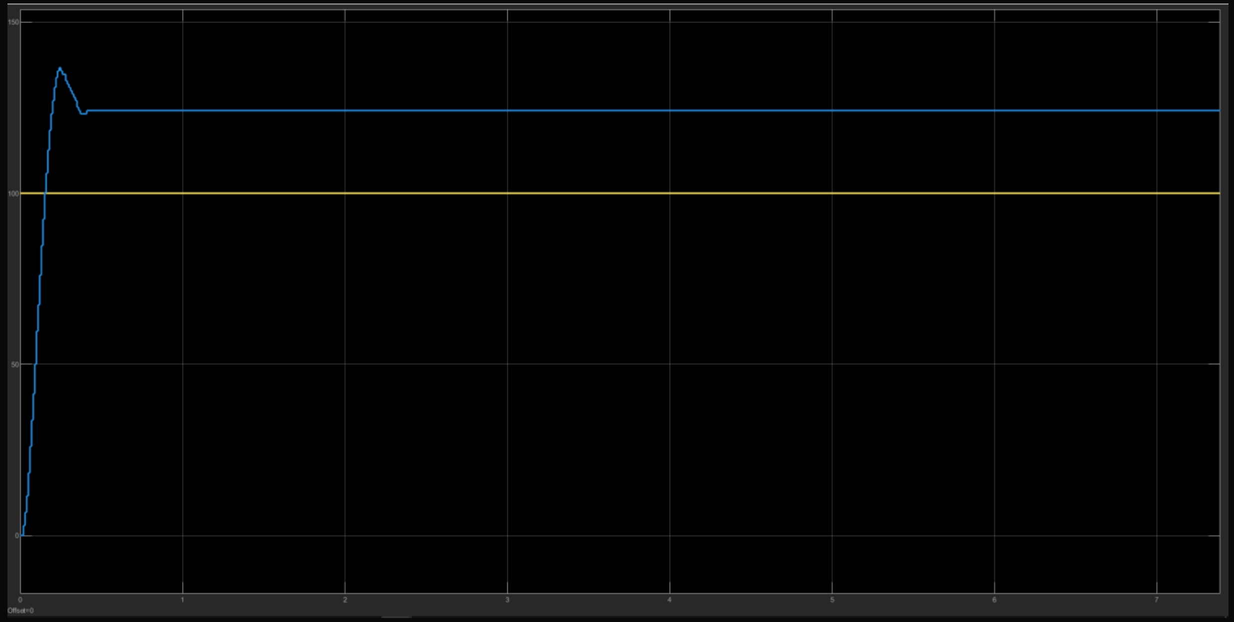


Parameters	Value
$e_{s.s.}$	24
M_p	0
t_p	0
t_s	0.43 sec.
t_r	0.29 sec.
t_c	0.24 sec.
t_d	0.2 sec.

Results

Connection and output of dc Motor Num. 2 (600RPM)





Parameters	Value
$e_{s.s.}$	24
M_p	9.6%
t_p	0.245 sec.
t_s	0.19 sec.
t_r	0.12 sec.
t_c	0.09 sec.
t_d	0.4 sec.

Tunning

Calculate Mathematical Model for DC Motor:

Speed

$$\boxed{\frac{\omega(s)}{V_a(s)} = \frac{K_T}{(R_a+L_a s)(Js+b)+K_b K_T}}$$

$$= \frac{K_T}{L_a J s^2+(R_a J+L_a b)s+(R_a b+K_b K_T)}$$

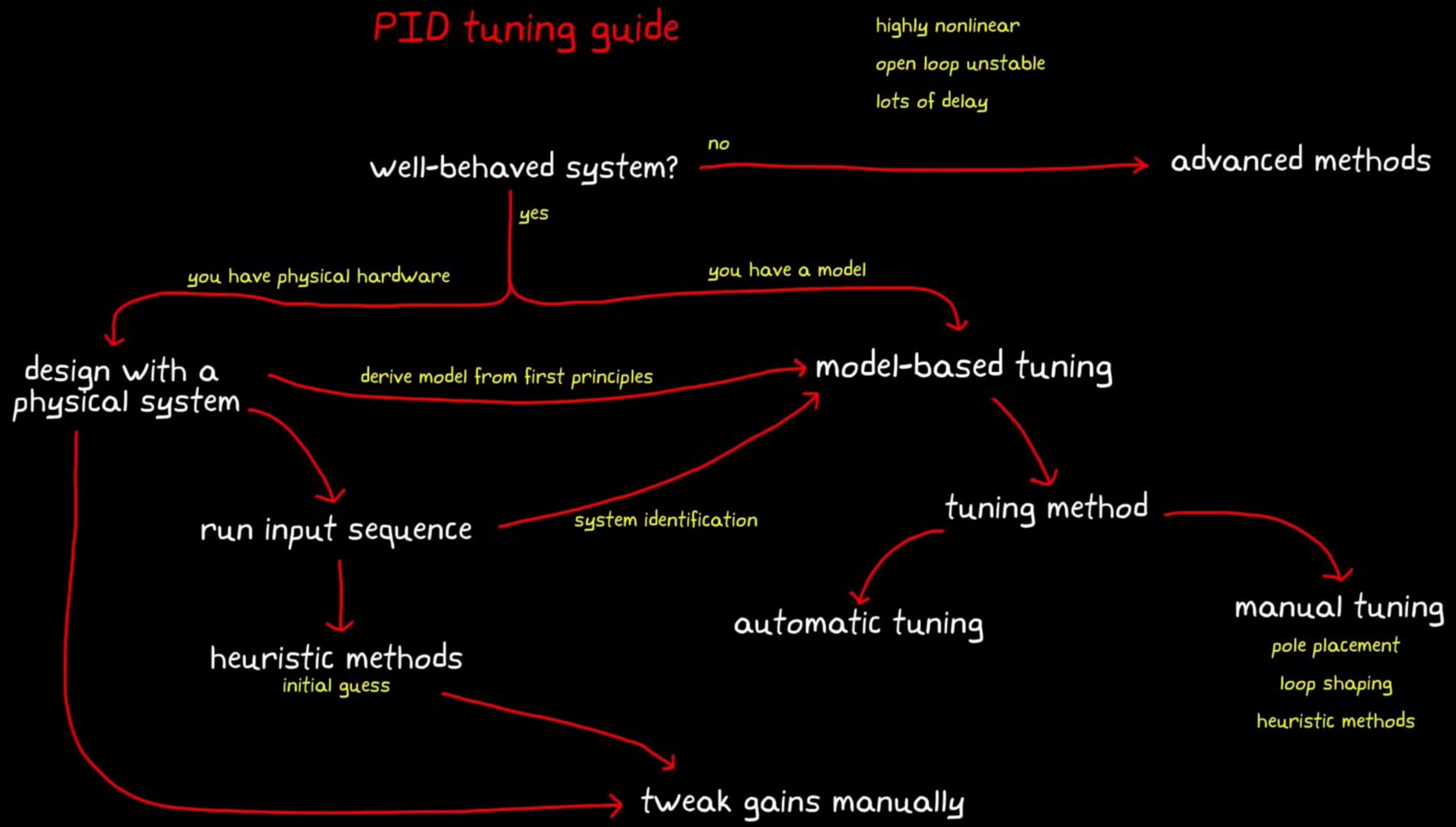
Where:
T_m: Motor torque.
K_T: Torque constant.
I_a: Armature current.
V_a: Input voltage.
R_a: Armature resistance.
L_a: Armature inductance.
V_b: Back electromotive force (EMF).
K_b: EMF constant.
ω: Angular velocity of rotor.
T_L: Load torque.
J: Rotating inertial measurement of motor bearing.
b: Fraction constant.

Position

- **Speed Control (Output =** ω**):**
  $\frac{\omega(s)}{V_a(s)}$ has no integrator → **Type 0 system.**
- **Position Control (Output =** θ**):**
  $\boxed{\frac{\theta(s)}{V_a(s)} = \frac{1}{s} \cdot \frac{k_t}{(L_a s+R_a)(Js+b)+k_e k_t}}$ includes one integrator → **Type 1 system.**

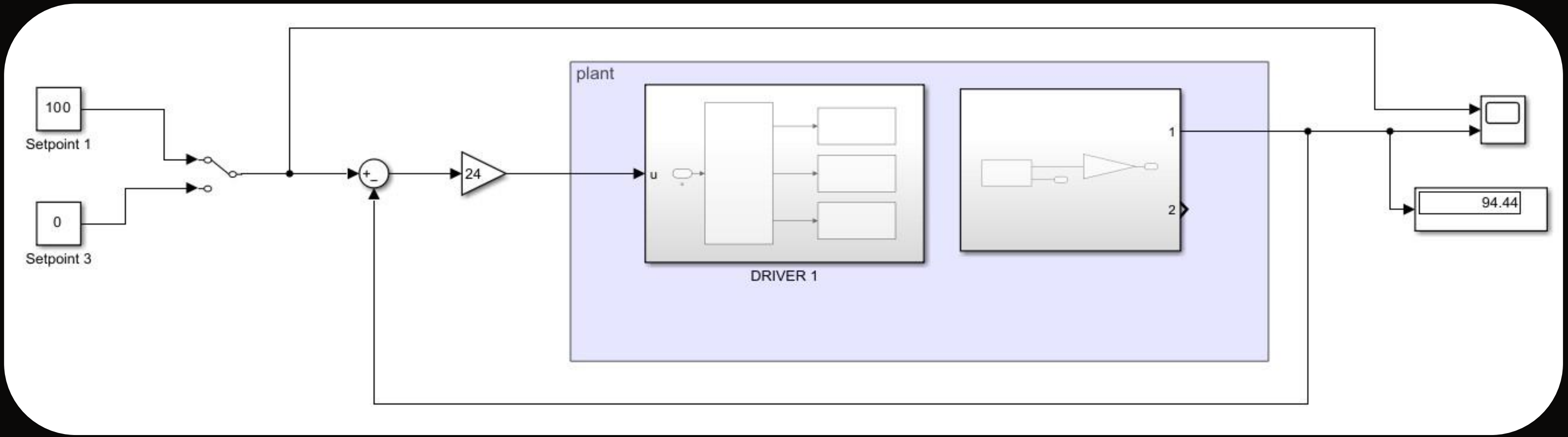
So, we chose PID Controller type

PID tuning guide



Tunning

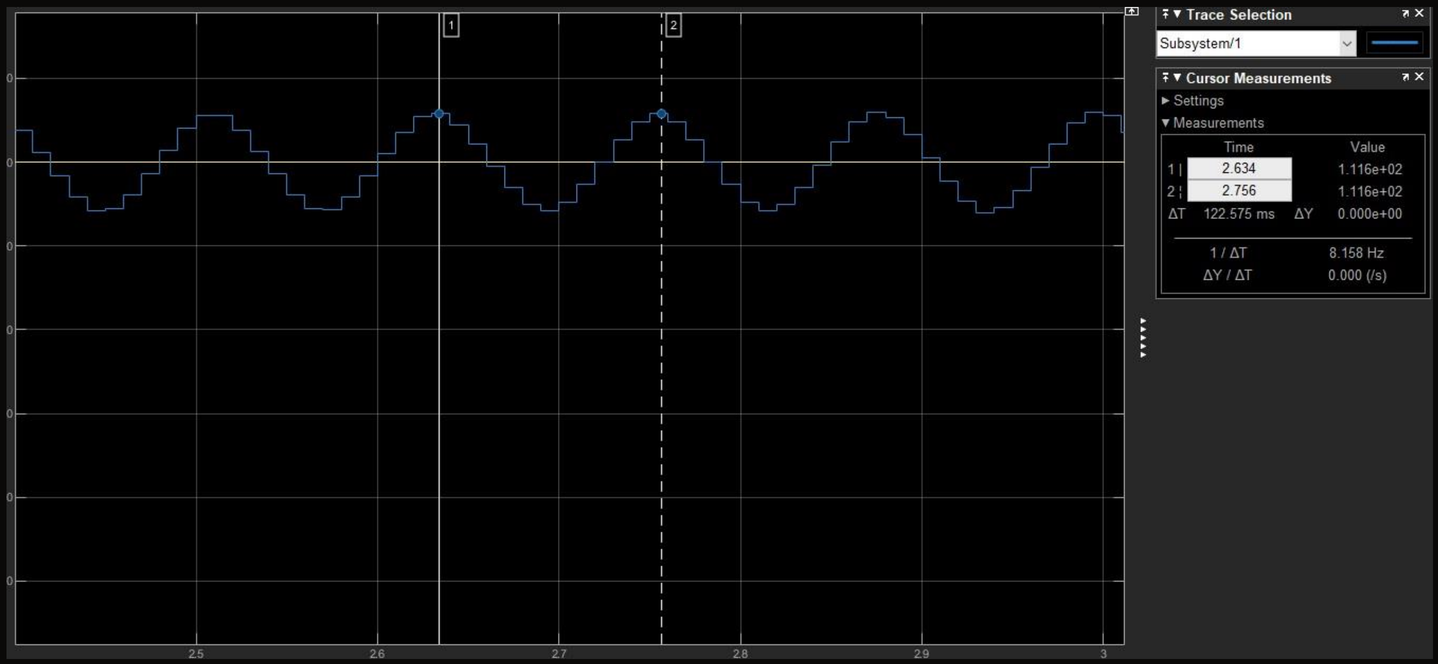
Using Ziggler & Nichol 2nd method
For DC Num. 1 (200RPM)



Kcr = 24 & Pcr = 0.122

Table 1 Ziegler–Nichols Tuning Rule Based on Critical Gain K_{cr} and Critical Period P_{cr} (Second Method)

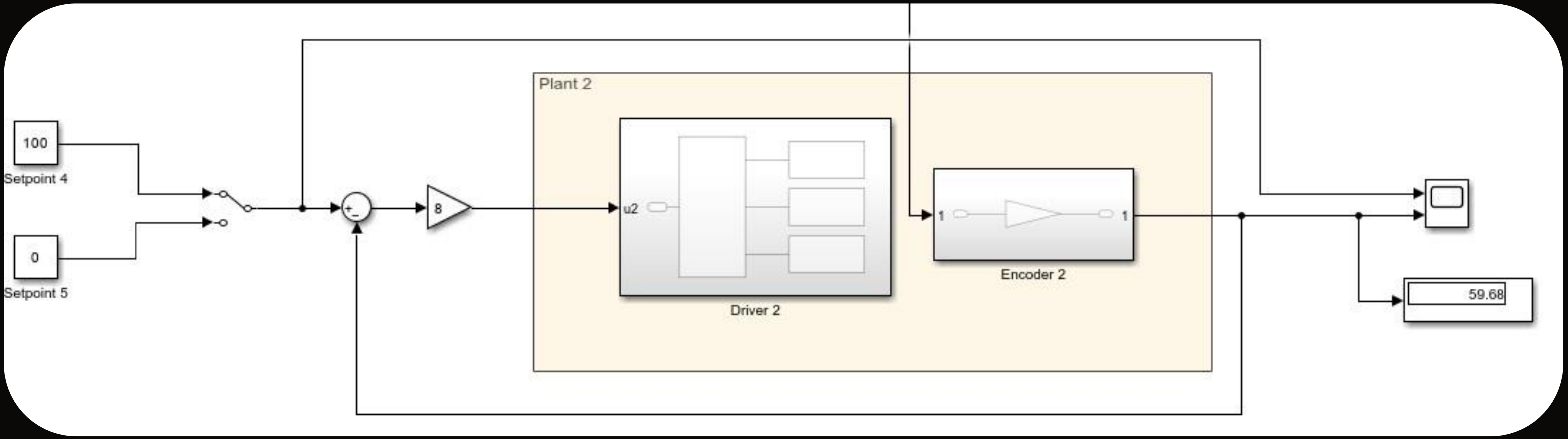
Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$



Parameters	Value
Kp	14.4
Ki	16.3
Kd	0.02

Tunning

Using Ziggler & Nichol 2nd method
For DC Num. 2 (600RPM)

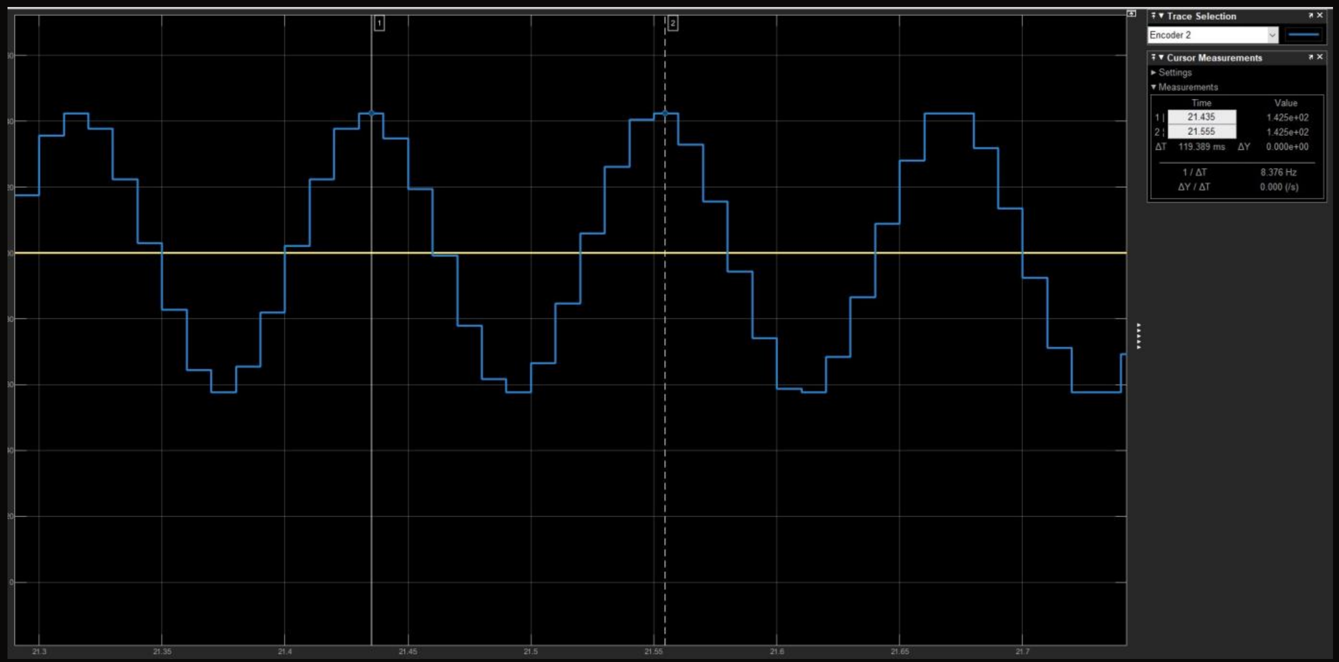


Kcr = 8 & Pcr = 0.119

Table 1 Ziegler–Nichols Tuning Rule Based on Critical Gain K_{cr} and Critical Period P_{cr} (Second Method)

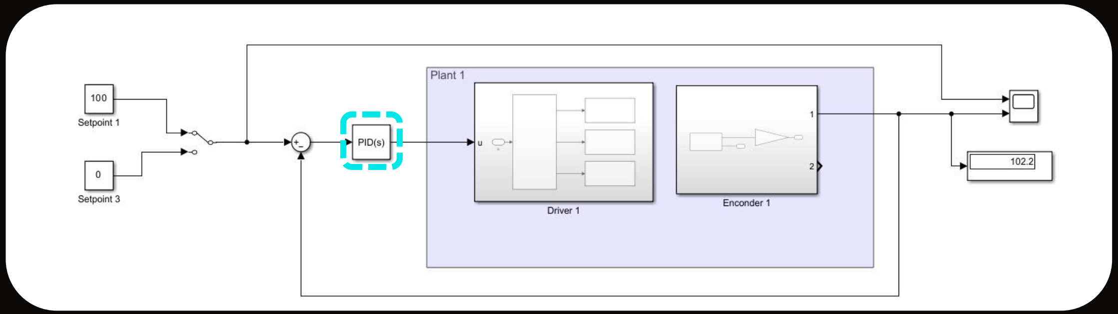
Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Parameters	Value
Kp	4.8
Ki	16.8
Kd	0.015

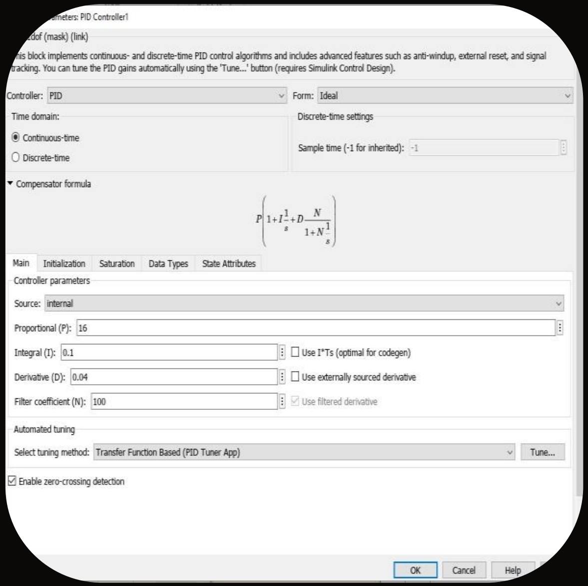


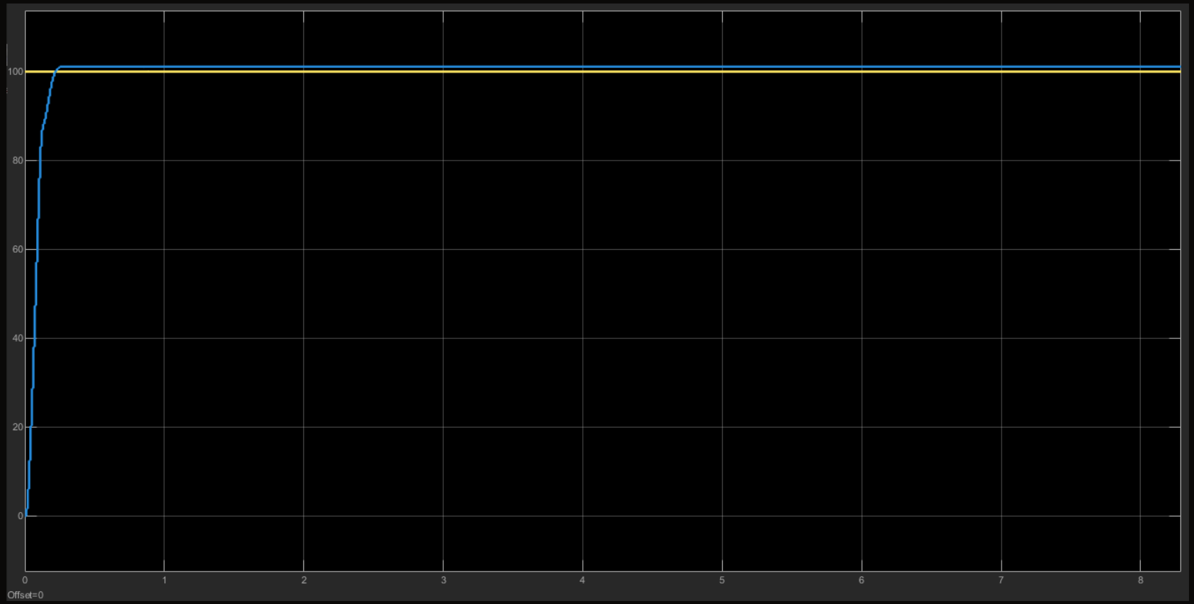
Results after Controller

Output of DC Num. 1 (200RPM) after Controller:



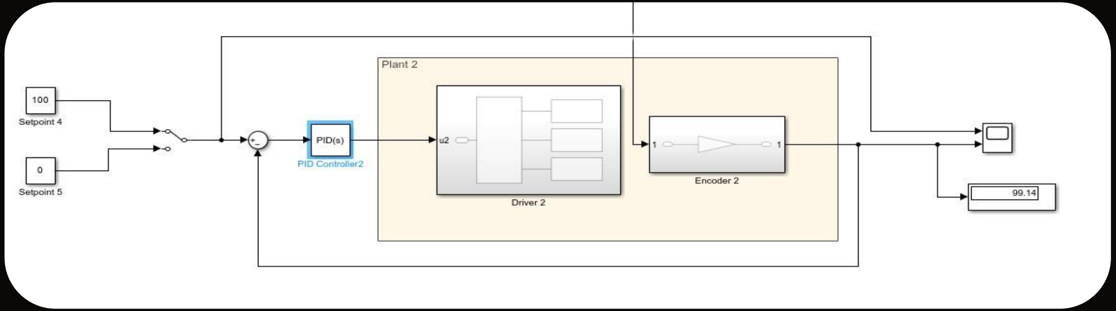
Parameters	Value
$e_{s.s.}$	1.1
Mp	0%
tp	0 sec.
ts	0.21 sec.
tr	0.13 sec.
tc	0.09 sec.
td	0.08 sec.



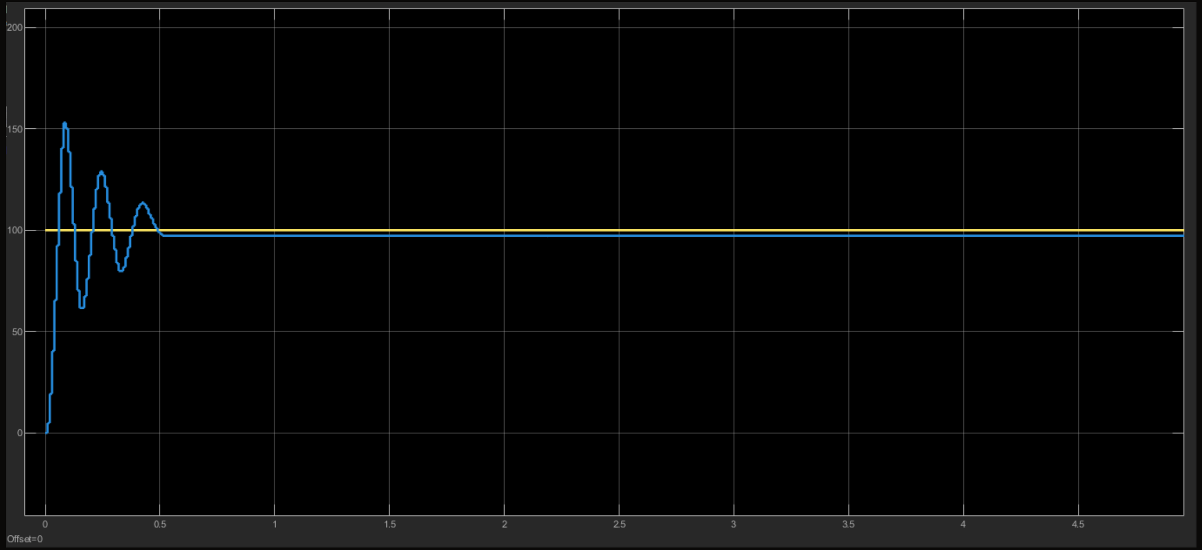


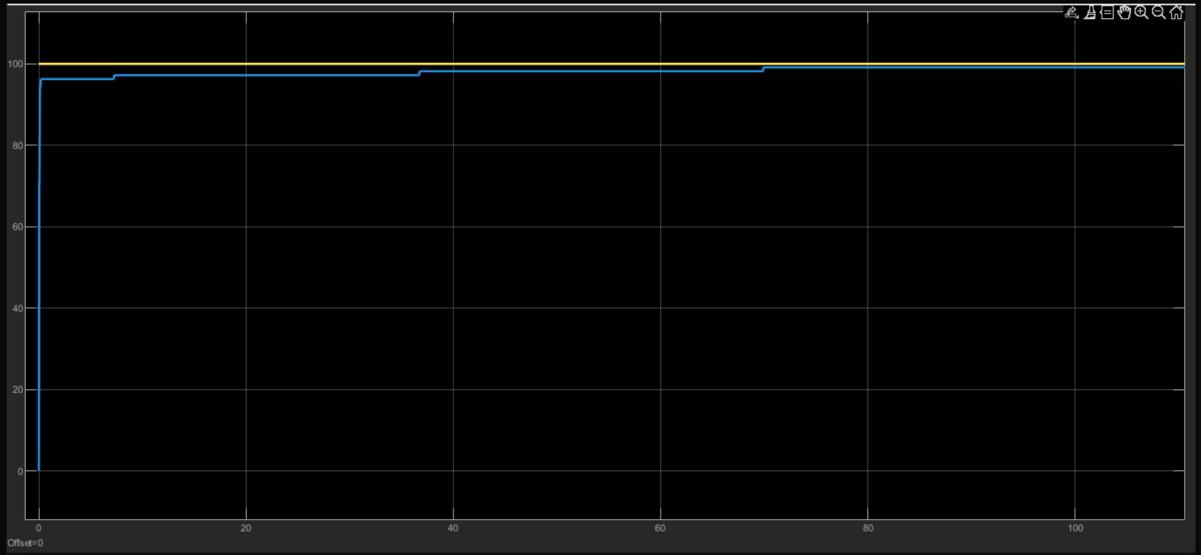
Results after Controller

Output of DC Num. 2 (600RPM) after Controller:

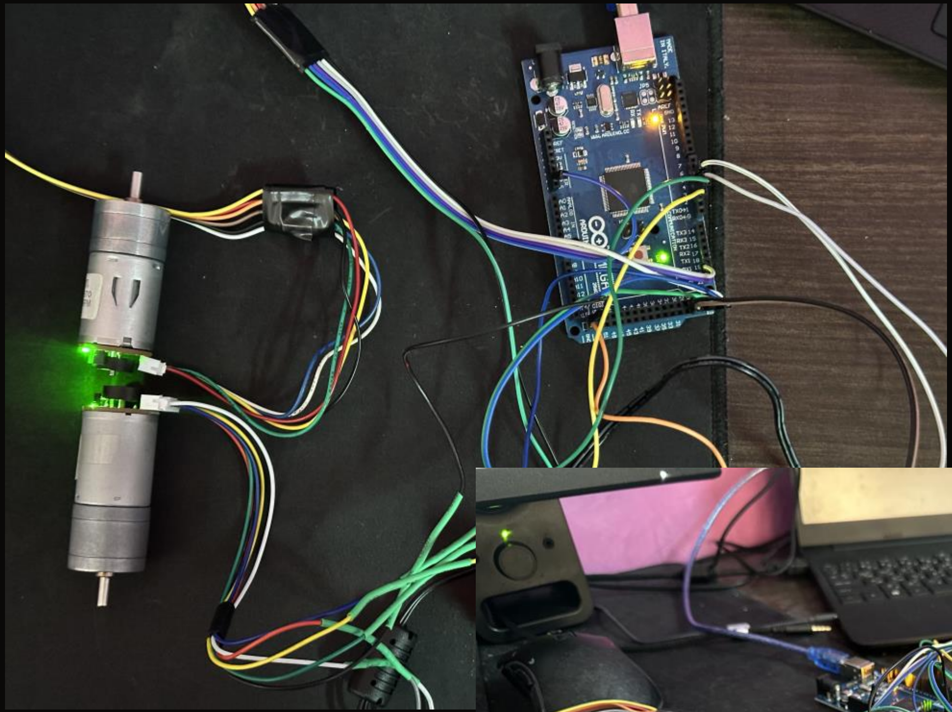


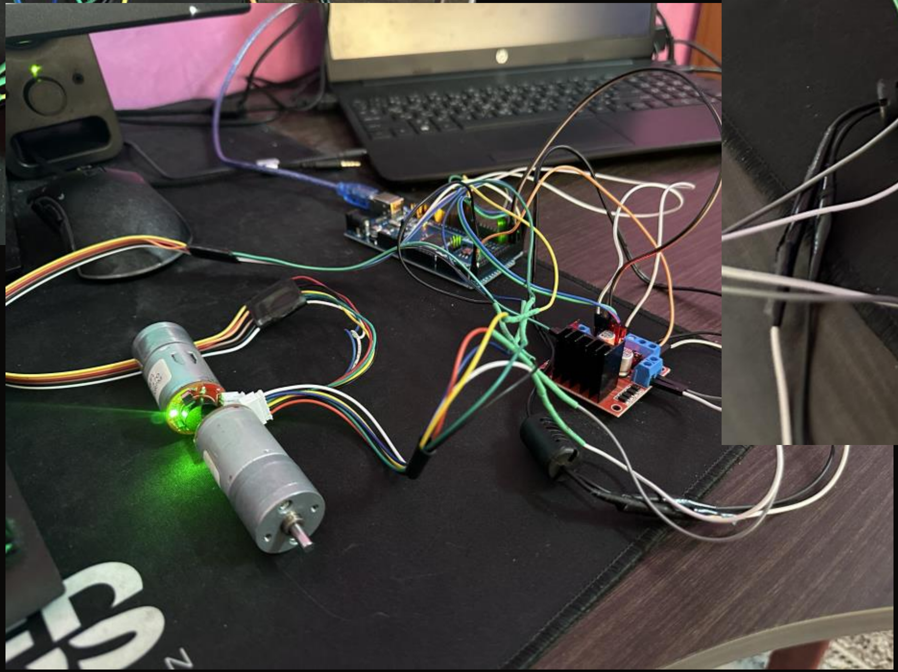
Parameters	Value
$e_{s.s.}$	0.86
M_p	0%
t_p	0 sec.
t_s	7.26 sec.
t_r	0.09 sec.
t_c	0.05 sec.
t_d	0.04 sec.

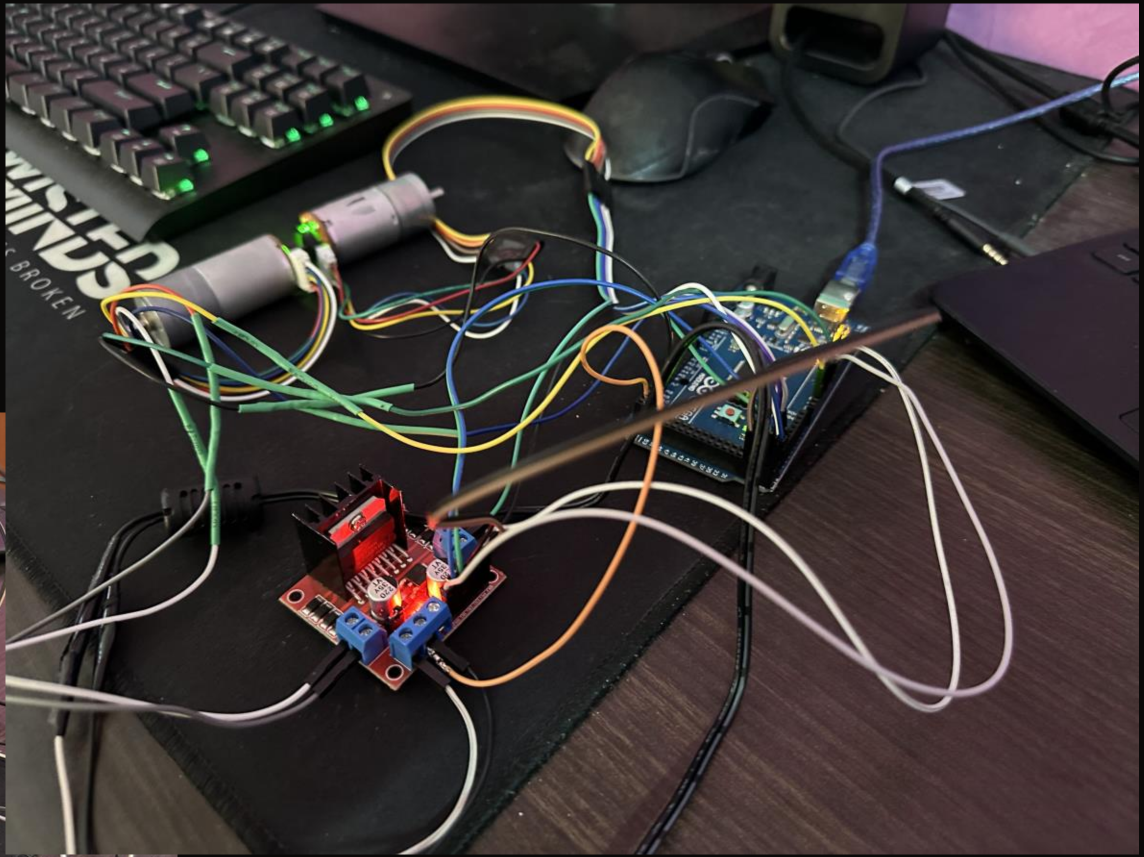




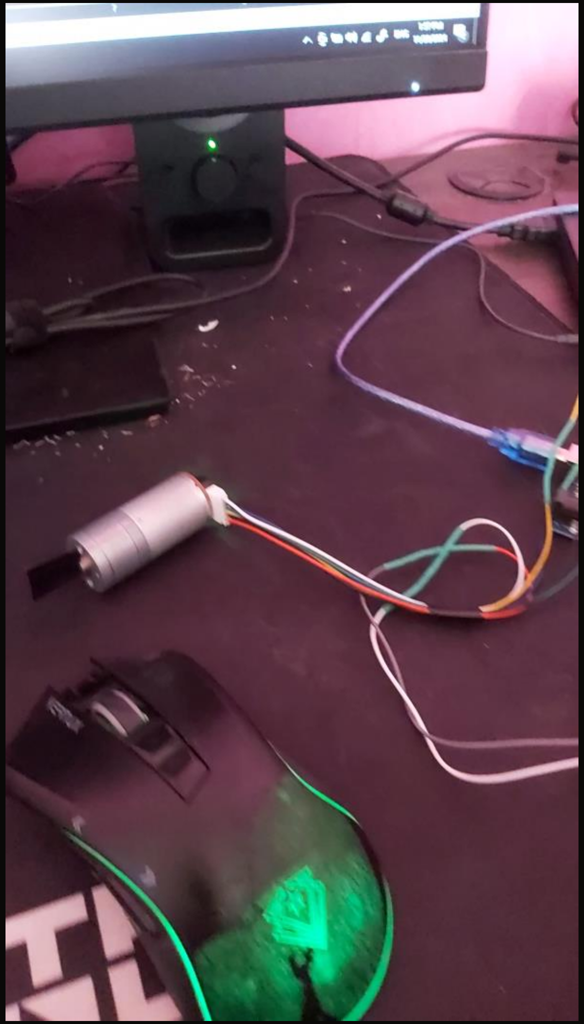
Demonstration

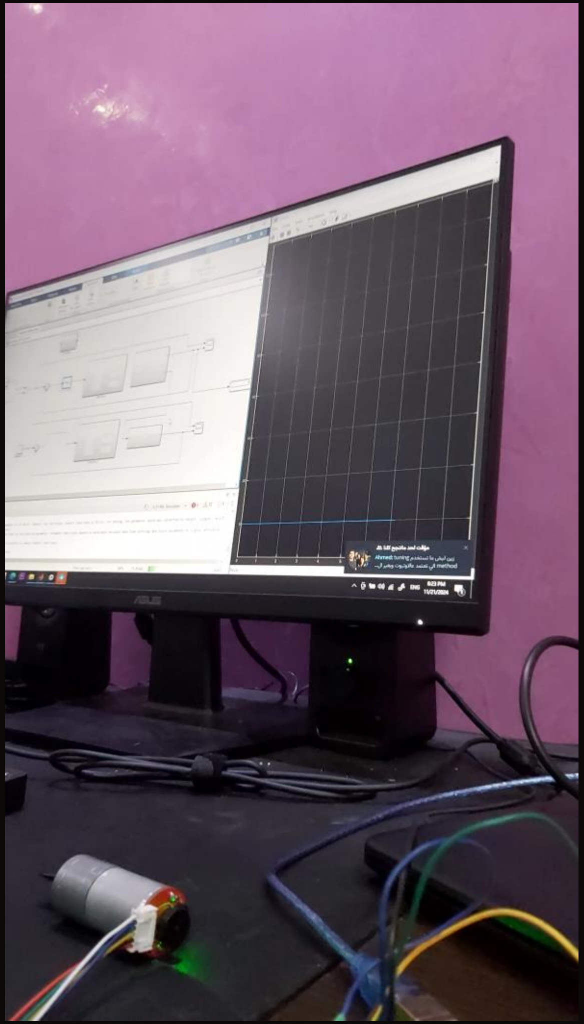


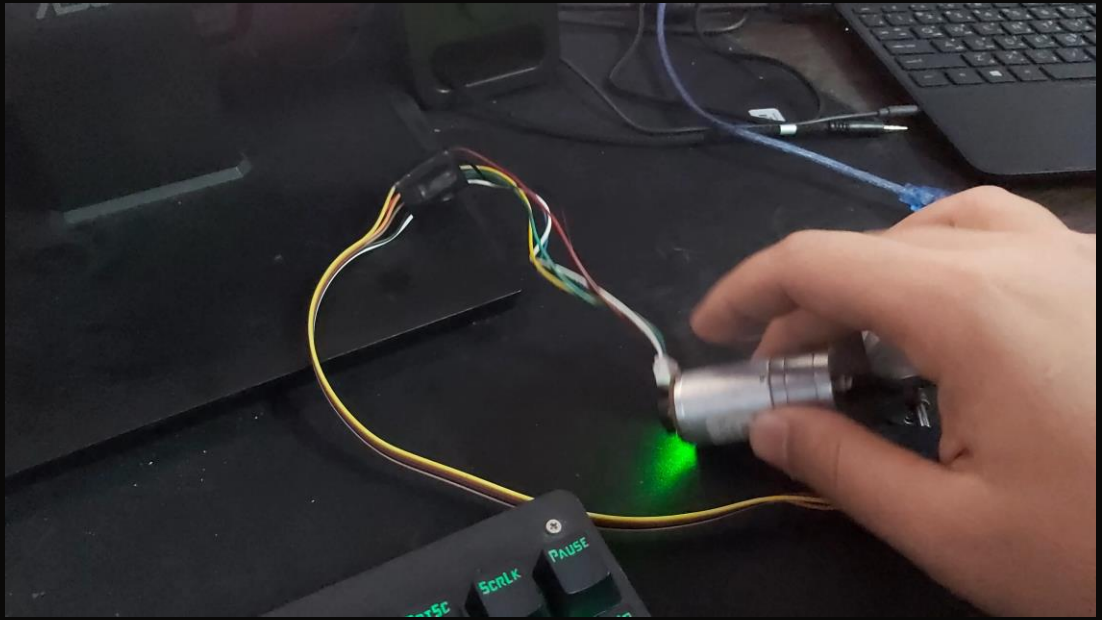




Demonstration







Challenges & Solutions

Challenge 1: We Don’t have parameter of Transfer function so,

- We can’t simulate the system
- We can’t apply all Tunning method
- The possibility of system failure increases.

Solution: Calculate Parameters manually

$$\frac{\omega(s)}{V_a(s)} = \frac{k}{(Js+B)(Ls+R)\ +k^2} \quad (8)$$

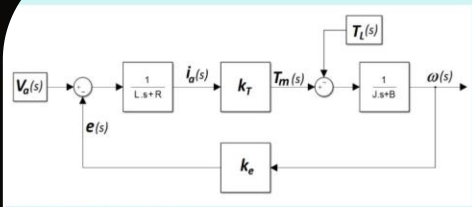


Fig.4 - DC Motor Transfer Function Diagram

Electrical Parameters of Armature

- The electrical parameters of armature were measured using the HIKARI HX-120 LCR digital meter:
  - Armaure resistance = 4.98 Ω
  - Armaure inductance = 3.8 mH

Torque and Back-EMF Constants

- Two JGA25-370 were coupled together. One of them was connected to L298N Bridge, controlled by the Arduino, in way that the speed could be varied. The armature voltage was measured in the second one. The results are presented below:

Table II - Speed and Armature Voltage Measurements

Speed (rps)	Speed (rad/s)	Armature Voltage (V)
2.07	13.0	7.5
1.82	11.44	6.55
1.52	9.55	5.51
1.26	7.92	4.4

- According to (4), the Table II data should represent a linear function crossing the origin, but considering the linear regression of these points, we can see some deviation. As it will be shown later, the lower the speed, the lower the speed measurement precision. Considering the highest speed, we have:

$$k_e = 0.577V/(rad/s) \quad (9)$$

o determine the torque constant, the L298N bridge was connected to a 12V power supply, and the motor was driven with the maximum duty cycle, resulting in V_a=10.6V, due to the saturation voltage over the bridge transistors, I_a=106mA and ω=2.78 rps

$$\frac{w(s)}{V_a(s)} = \frac{K_T}{L_aJs^2 + (R_aJ + L_ab)s + (R_ab + K_bk_t)}$$

Where:

- Va = armature voltage (v)
- Ra = armature resistance Ω
- La = armature inductance (H)
- Ia = armature current (A)
- Eb = back emf (V)
- w=angular speed (rad/s)
- Tm =motor torque (Nm)
- θ =angular position of rotor shaft (rad)
- ginJmjm = rotor inertia
- (kg m2m2)Bm = viscous friction coefficient (Nms/rad)
- KT = torque constant (Nm/A)
- Kb =back.emf constant (Vs/rad)

pecification:

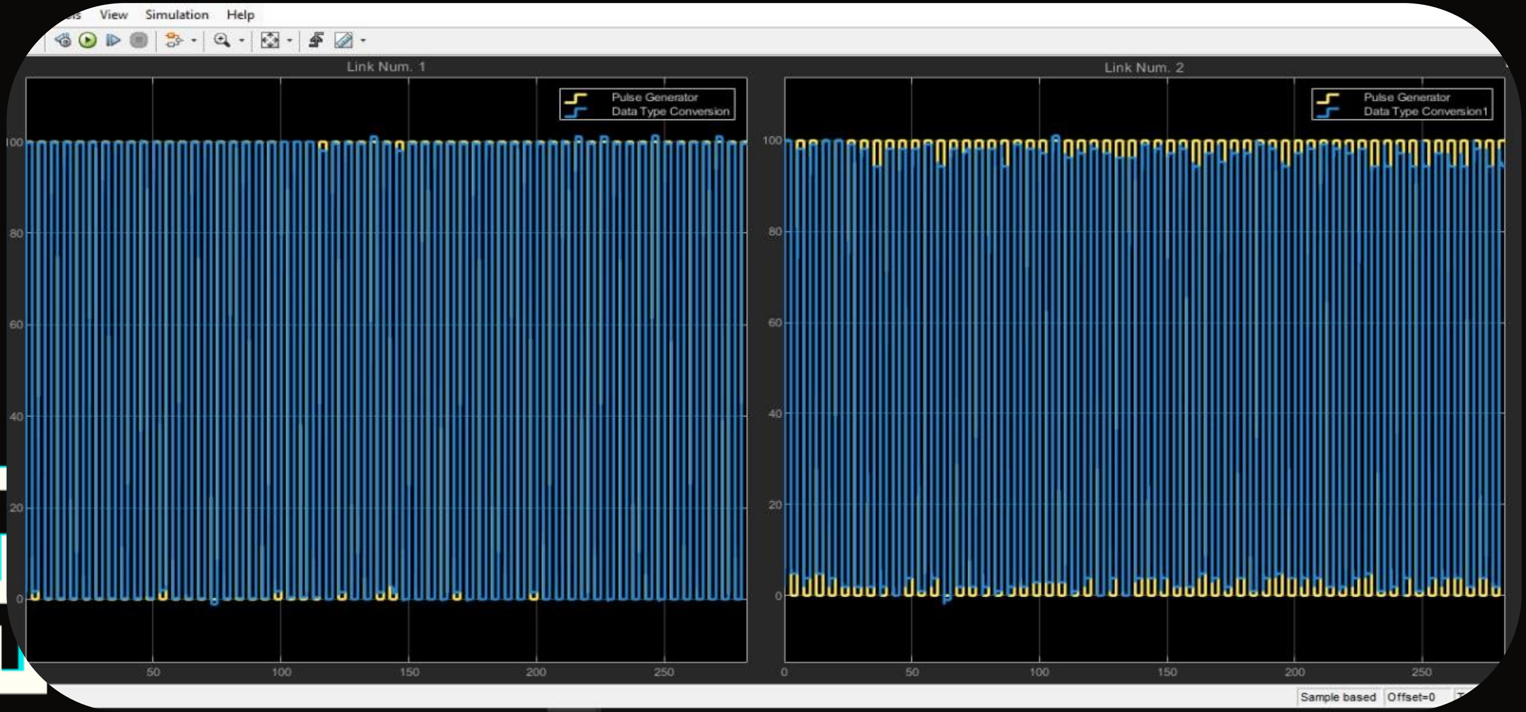
Voltage V	No-load		Maximum efficiency pointed				Blockage	
	speed r/min	electric current A	speed r/min	electric current A	Torque Kg.cm	Power W	Torque Kg.cm	electric current A
6	190	0.2	133	0.5	0.75	1.1	4.0	2.1
12	350	0.1	245	0.65	1.4	2.4	5.2	2.2

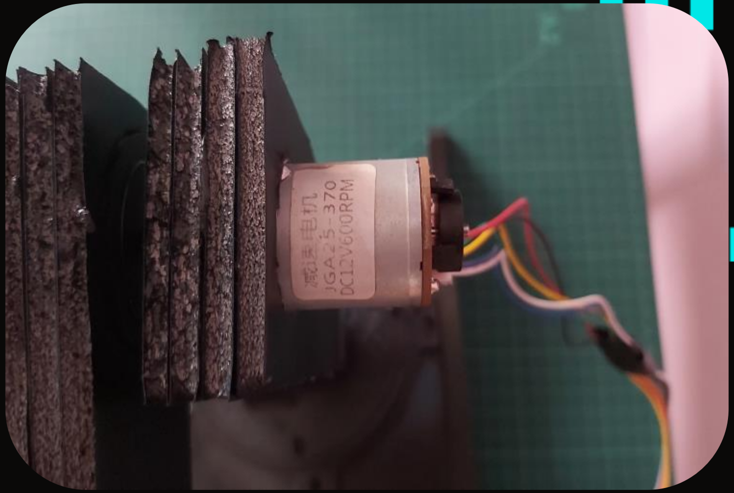
Challenges & Solutions

Challenge 2: The motor is not widely available in the Iraqi market

- Incompatibility between the two motors
- More effort
- More time

Solution: Buy the motors in one Time



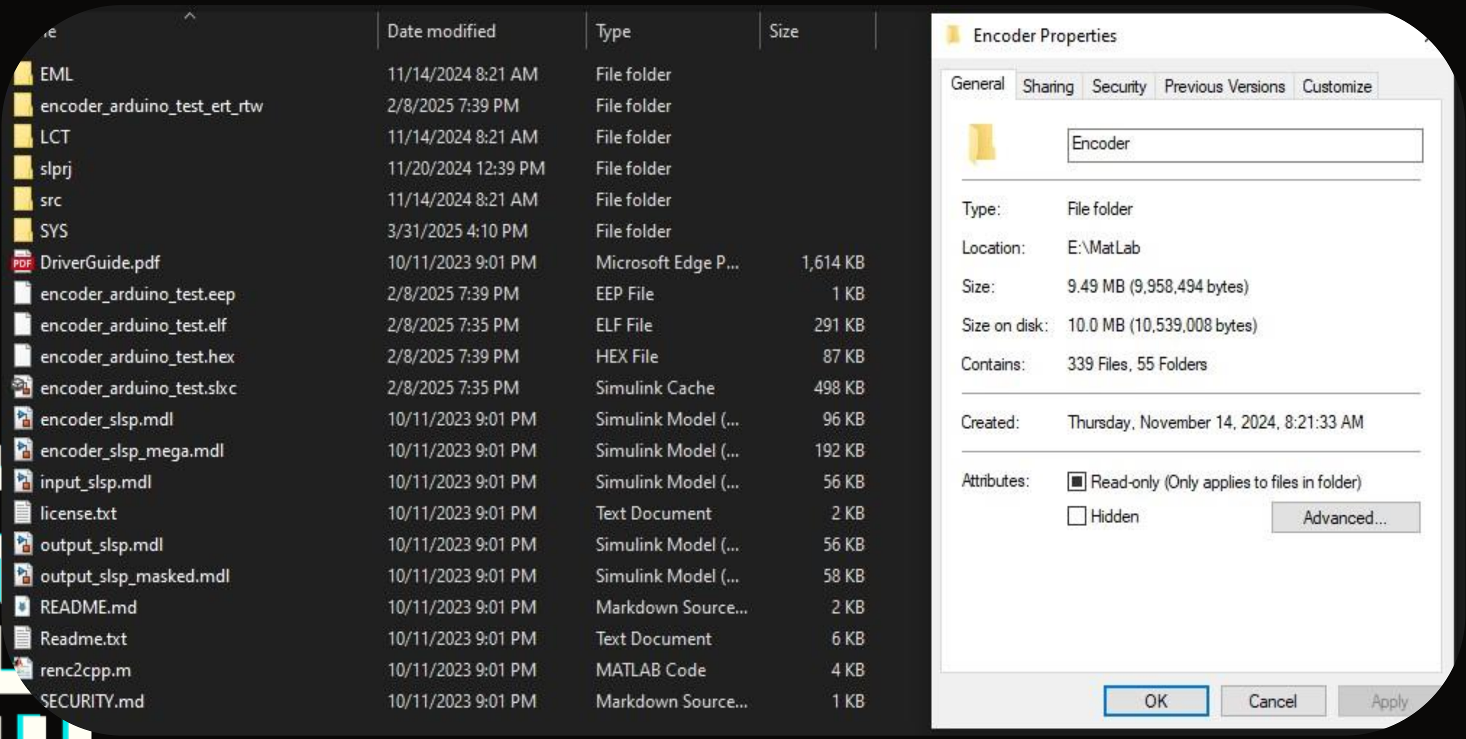


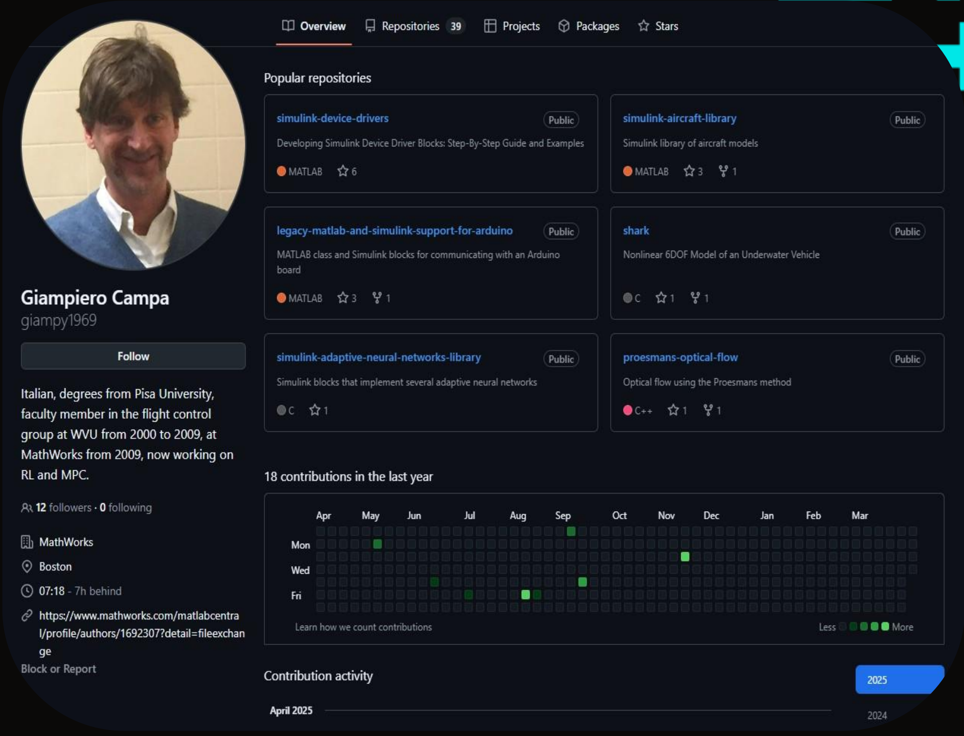
Challenges & Solutions

Challenge 3: Hard to read from encoder

- You need to know how to deal with timers and interrupts in Arduino
- You must learn how to link the cpp file with Simulink
- If you use two encoder so, you need four interrupt pin

Solution: Use Device Driver library and Arduino Mega



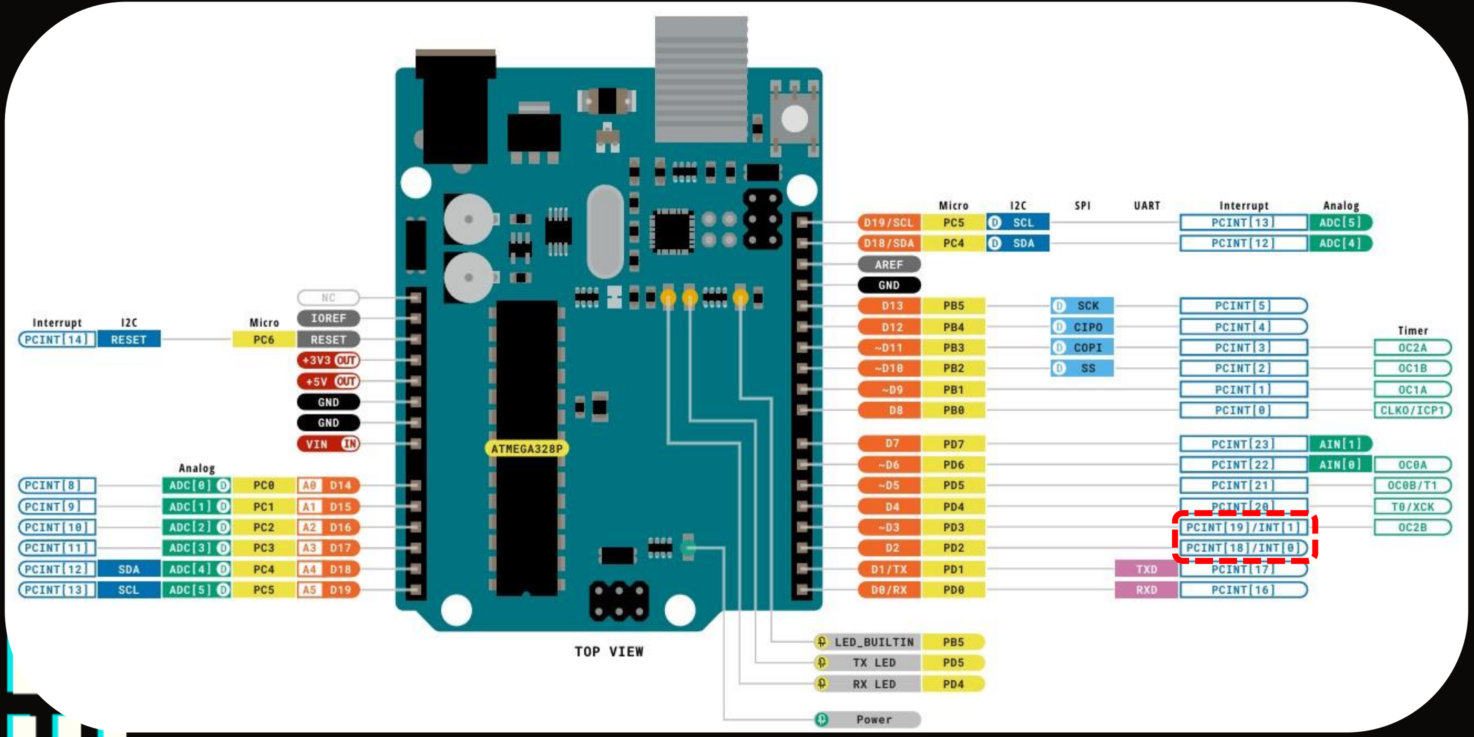


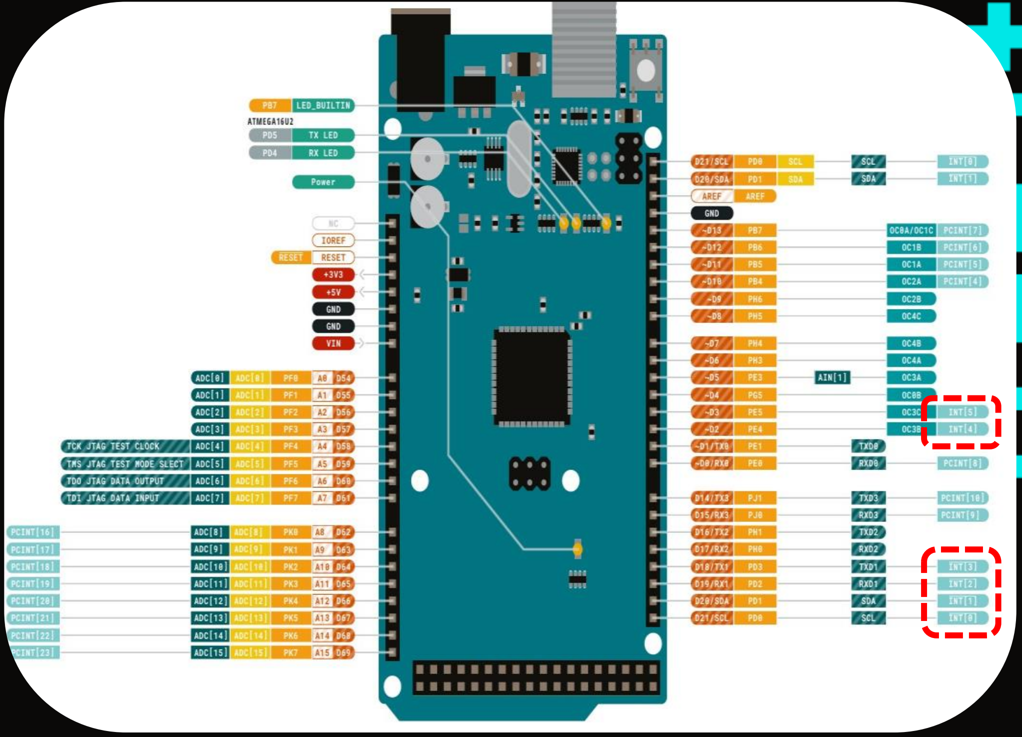
Challenges & Solutions

Challenge 3: Hard to read from encoder

- You need to know how to deal with timers and interrupts in Arduino
- You must learn how to link the cpp file with Simulink
- If you use two encoder so, you need four interrupt pin

Solution: Use Device Driver library and Arduino Mega





Challenges & Solutions

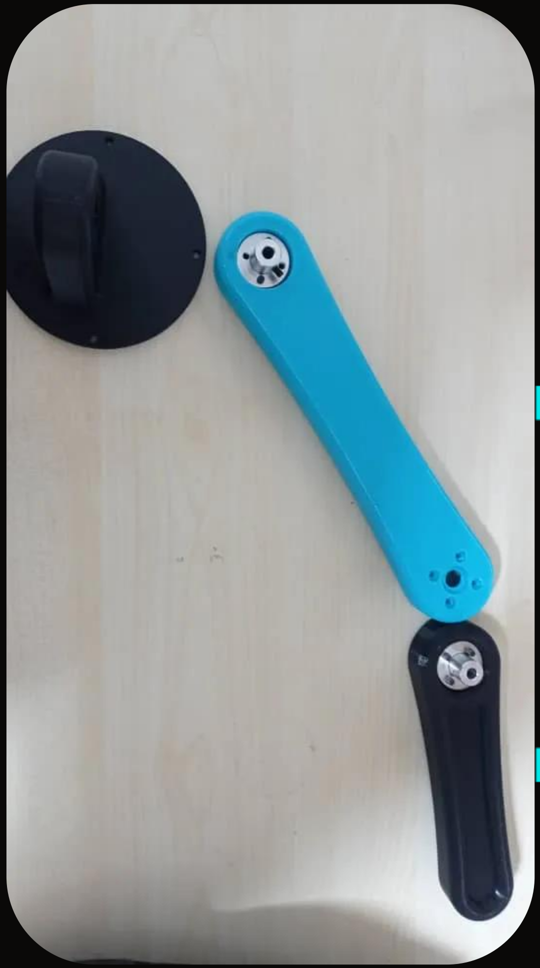
Challenge 4: The structure

Challenges & Solutions

Challenge 4: The structure

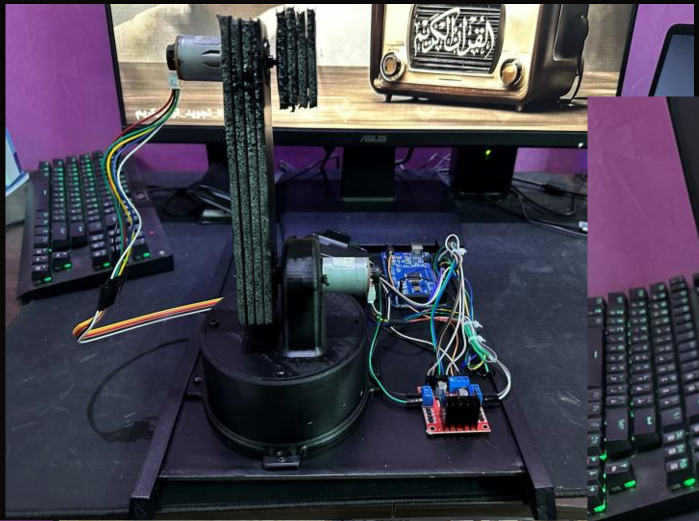
Challenges & Solutions

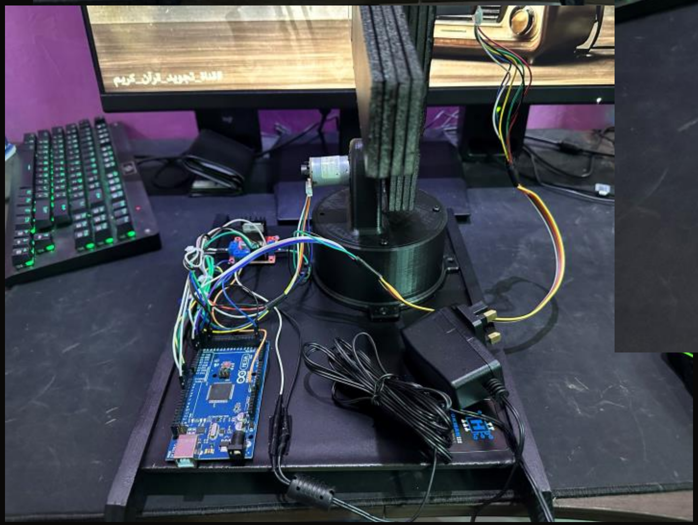
Challenge 4: The structure

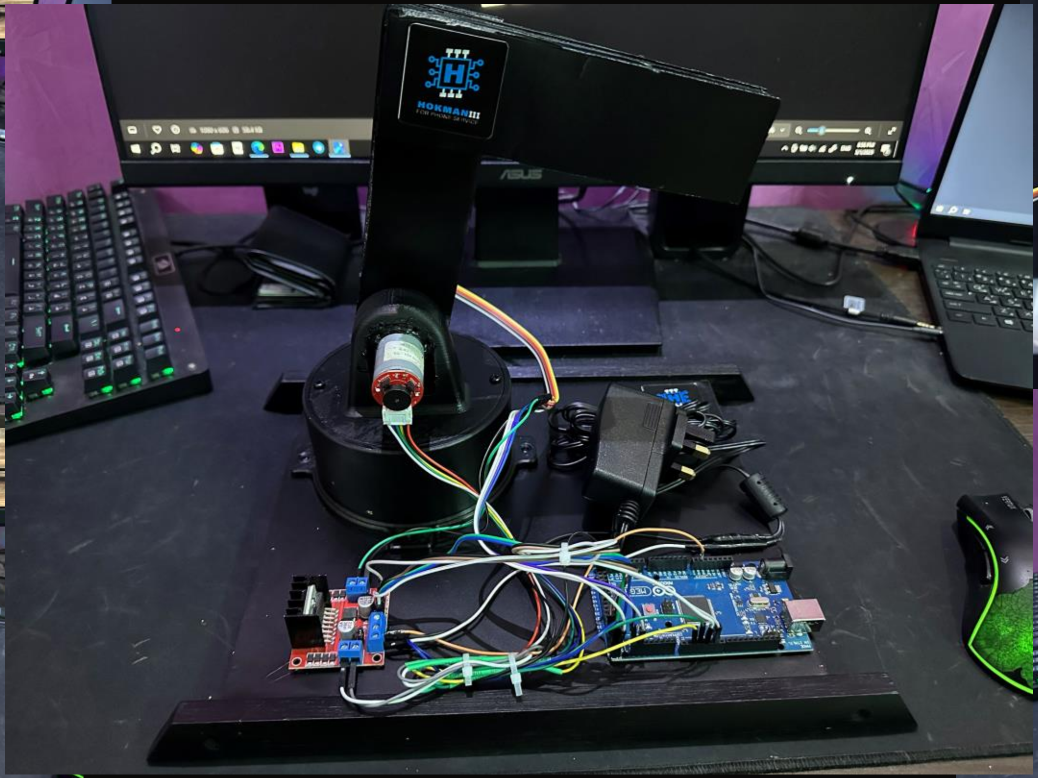


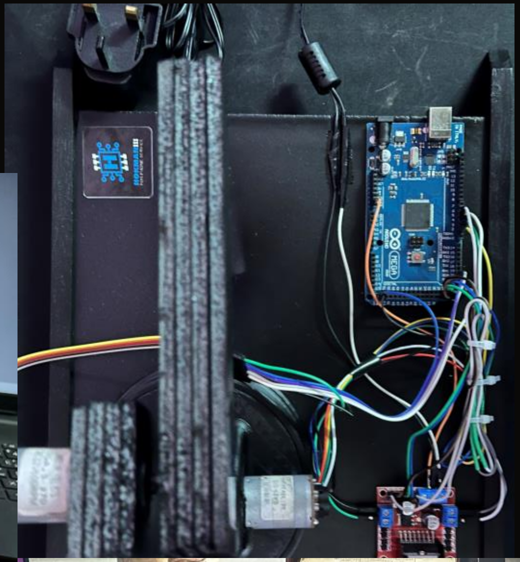
Challenges & Solutions

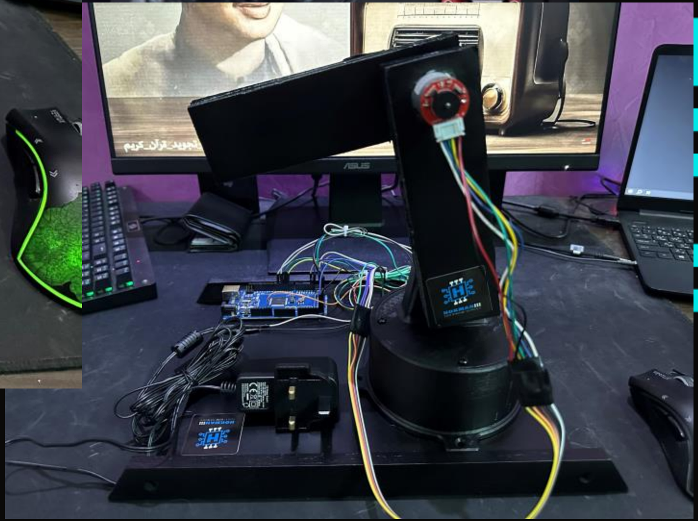
Challenge 4: The structure









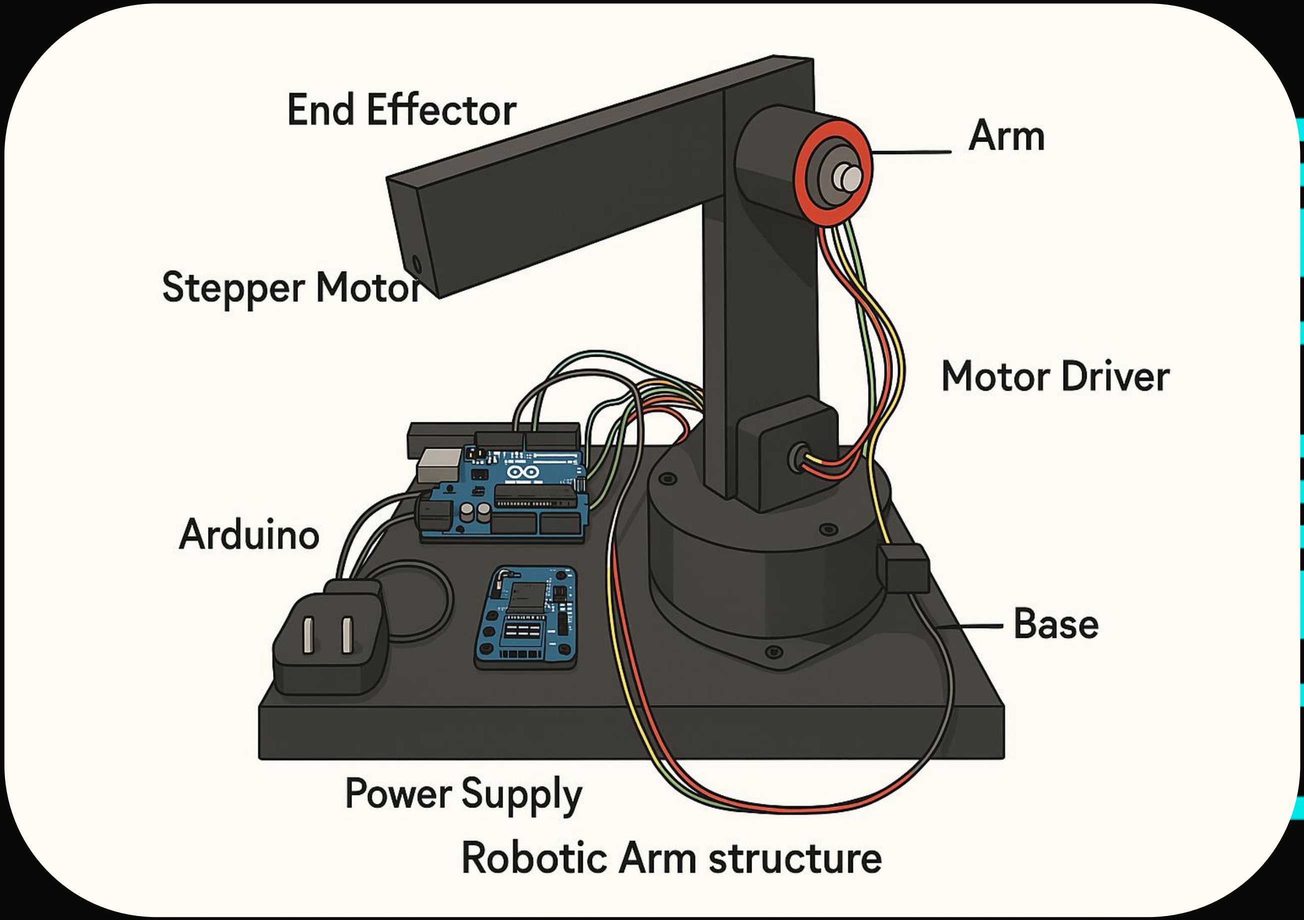


Future Improvements

- Adding two additional joints to function as a full arm
- Making control automatic without manually input
- Printing a new structure compatible with the project
- Using artificial intelligence to find controller parameters
- Using unconventional (Adaptive) controller type

Conclusion

- Despite the difficulties I faced throughout the project, I was able to successfully develop a robotic arm simulator using two DC motors. By implementing a PID control system, I achieved precise and reliable movement of the motors, ensuring accurate angle adjustments. This project not only demonstrated the practicality of control systems in robotics but also provided valuable insights into overcoming challenges in real-world applications



Reference's

- www.mathworks.com/requirement
- L298N DC Motor Driver – How It Works
- DC MOTOR with encoder – How It Works
- MATLAB Support Package for Arduino Hardware
- Simulink Support Package for Arduino Hardware
- Device Drivers
- Mathematical Model for DC Motor
- Calculate Parameters of DC manually
- JGA25-370 DC MOTOR Datasheet
- Arduino Mega datasheet
- Arduino Uno datasheet

Thank To

Sajjad Monther
Designed the visual identity

Mustafa Jamal
Printed the outer structure

Dr. Ekhlas Karam
Project Supervisor

Ahmed Hazam
Help me to solve problems

Hussien Hamza
Help me to solve problems

Al-Qassam Team
First and last supporter